

PROPERTY OF
ANL-W Technical Library

THE SYSTEMS ANALYSIS LANGUAGE TRANSLATOR (SALT): USER'S GUIDE

by

Howard K. Geyer and Gregory F. Berry



FOSSIL ENERGY PROGRAM

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Operated by THE UNIVERSITY OF CHICAGO

for the U. S. DEPARTMENT OF ENERGY

under Contract W-31-109-Eng-38

Argonne National Laboratory, with facilities in the states of Illinois and Idaho, is owned by the United States government, and operated by The University of Chicago under the provisions of a contract with the Department of Energy.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Printed in the United States of America
Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A08
Microfiche copy: A01

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue, Argonne, Illinois 60439

ANL/FE-85-3

THE SYSTEMS ANALYSIS LANGUAGE TRANSLATOR (SALT):
USER'S GUIDE

by

Howard K. Geyer* and Gregory F. Berry

Energy and Environmental Systems Division

January 1985

work sponsored by

U.S. DEPARTMENT OF ENERGY
Morgantown Energy Technology Center

*Engineering Division.

CONTENTS

ABSTRACT	1
1 INTRODUCTION	1
1.1 Overview	1
1.2 Examples	2
1.3 Job-Control Language	12
2 SALT LANGUAGE	13
2.1 PROCESS Statement	13
2.2 SYSBEG and SYSEND Statements	18
2.3 VARY Statement	18
2.4 CONSTRAIN Statement	19
2.5 MINIMIZE Statement	19
2.6 SWEEP Statement	20
2.7 INTEGRATE Statement	20
2.8 STEADY Statement	21
2.9 CONTROL Statement	21
2.10 DATA Statement	22
2.11 SWITCH Statement	22
2.12 PLI Statement	23
3 MODELS	24
3.1 Model and Flow Variables	24
3.2 Initialization of Properties Calculations	26
3.3 Demand-Type Models	27
3.4 Flows Used in Steady-State Models	28
4 STEADY-STATE MODELS	29
4.1 Generic Combustor/Gasifier Model	29
4.2 Generic System-Component Models	30
4.2.1 Compressor Model	30
4.2.2 Deaerator Model	31
4.2.3 Flash Model	31
4.2.4 Feedwater-Heater Model	32
4.2.5 Heater Model	33
4.2.6 Heat-Exchanger Model	33
4.2.7 Flow-Initiator Model	35
4.2.8 Flow-Mixer Model	37
4.2.9 Steam-Condenser Model	37
4.2.10 Steam-Drum Model	37
4.2.11 Flow-Splitter Model	37
4.2.12 Pump Model	38
4.2.13 Fuel-Flow-Initiator Model	38
4.2.14 Fuel-Dryer Model	39
4.2.15 Nozzle Model	39
4.2.16 Diffuser Model	39
4.2.17 Stack Model	40

CONTENTS (Cont'd)

4.3	Turbine Models	40
4.3.1	Steam-Turbine Model	40
4.3.2	Gas-Turbine Model	42
4.4	Fuel-Cell Models	43
4.4.1	Molten-Carbonate Fuel-Cell Model	43
4.4.2	Solid-Oxide Fuel-Cell Model	44
4.4.3	Phosphoric Acid Fuel-Cell Model	44
4.5	Magnetohydrodynamic-Generator Model	45
4.6	Component Models for Liquid-Metal Systems	46
4.6.1	Liquid-Metal Pipe Model	46
4.6.2	Liquid-Metal Nozzle Model	47
4.6.3	Liquid-Metal Diffuser Model	47
4.6.4	Liquid-Metal Magnetohydrodynamic-Generator Model	48
4.7	Two-Component Liquid/Gas Separator Model	48
4.8	Two-Phase Component Models	49
4.8.1	Two-Phase Mixer Model	49
4.8.2	Two-Phase Nozzle Model	50
4.8.3	Two-Phase Diffuser Model	51
4.9	System Model	51
5	EXAMPLES USING STEADY-STATE MODELS	53
5.1	Simple System Configurations	53
5.1.1	Simple Steam-Flow System	53
5.1.2	Inclusion of a System Model	54
5.1.3	Inclusion of a Demand Model Constraint	55
5.1.4	Inclusion of User-Imposed Constraints	57
5.1.5	Inclusion of an Optimization Problem	59
5.1.6	Inclusion of a Parameter Sweep	60
5.1.7	Inclusion of a Feedback Loop	61
5.2	Summary of the Input-Formation Process	63
5.3	Analysis of Power-Plant Systems	63
5.3.1	Fossil/Steam Power Plant	64
5.3.2	Open-Cycle Magnetohydrodynamic Power Plant	70
5.3.3	Solid-Oxide Fuel-Cell System	73
5.3.4	Liquid-Metal Magnetohydrodynamic System	77
6	FAILURE CAUSES AND CURES	83
6.1	Introduction	83
6.2	Failures Due to Mathematical Problems	83
6.2.1	Constraining Tasks	83
6.2.2	Optimization Tasks	87
6.3	Failures Due to Physical Problems	89
7	ADDING NEW MODELS AND FLOW TYPES	92
7.1	The Interface File	92
7.2	Adding New Models	94
7.3	Adding New Flow Types	95

CONTENTS (Cont'd)

REFERENCES	97
APPENDIX A: Job-Control Language for IBM System at ANL	99
APPENDIX B: Abbreviations for Key Words	103
APPENDIX C: Units Used in Salt Models	107
APPENDIX D: Fossil/Steam Power Plant	111
APPENDIX E: Open-Cycle Magnetohydrodynamic Power Plant	123
APPENDIX F: Solid-Oxide Fuel-Cell System	135
APPENDIX G: Liquid-Metal Magnetohydrodynamic Power Plant	145

FIGURES

1 Simple Steam Path	2
2 Simple Steam Path with Bypass Leg	3
3 Parallel-Path Steam System	4
4 System with a Multiple-Entry Model	5
5 Alternative Flow Arrangements for the Splitter Model	13
6 Model with One Pass-Through Flow, One Input Flow, and One Output Flow	14
7 Flow-Initiator Model	14
8 Simple Three-Model System	15
9 Simple Five-Model System	16
10 Model with Multiple Pass-Through Flows	16
11 Example Using a Multiple-Entry Model	17
12 Simple Steam-Plant System	55
13 Simple Steam-Plant System with Feedback Loop	61
14 Fossil/Steam Power Plant	65
15 Open-Cycle Magnetohydrodynamic Power Plant	71

CONTENTS (Cont'd)

16	Solid-Oxide Fuel-Cell System	74
17	Liquid-Metal Magnetohydrodynamic System	78

THE SYSTEMS ANALYSIS LANGUAGE TRANSLATOR (SALT): USER'S GUIDE

by

Howard K. Geyer and Gregory F. Berry

ABSTRACT

The Systems Analysis Language Translator (SALT), a systems-analysis and process-simulation computer code for steady-state and dynamic systems, can also be used for optimization and sensitivity studies. The SALT code uses sophisticated numerical techniques, including a hybrid steepest-descent/quasi-Newtonian multidimensional nonlinear equation solver, sequential quadratic programming methods as optimizers, and multistep integration methods for both stiff and nonstiff systems of equations. Based on a preprocessor concept, the code uses a language translator to allow the user great flexibility in specifying a systems-analysis problem using a mostly free format and user-defined labels. The code uses precompiled component models, several flow types, and numerous thermodynamic and transport property routines, including a gas chemical-equilibrium code. The SALT code has been used to study open-cycle and liquid-metal magnetohydrodynamic systems, fuel cells, ocean thermal energy conversion, municipal-solid-waste processing, fusion, breeder reactors, and geothermal and solar-energy systems.

1 INTRODUCTION

1.1 OVERVIEW

The Systems Analysis Language Translator (SALT) is a steady-state (see Refs. 1-7) and dynamic (see Ref. 8) system code that can analyze lumped-component systems of arbitrary configuration. As a steady-state code, SALT excels at performing nonlinearly constrained optimization studies and parametric studies and can establish all kinds of user-imposed system constraints. The code uses state-of-the-art problem-solving techniques, including hybrid steepest-descent/quasi-Newtonian equation solvers⁹ and sequential quadratic programming methods as optimizers.¹⁰ As a dynamic code, SALT uses multistep methods for both stiff and nonstiff systems and determines steady-state initial values using hybrid techniques.¹¹

The SALT code is a preprocessor that accepts input from two primary files, STRUCT and INTF, and generates a PL/I code representing a given system problem. The STRUCT file contains user-supplied data representing the system configuration for the specific problem to be analyzed, together with instructions defining system constraints, objective functions, parameter sweeps, etc. The INTF file contains information needed

at interfaces with the system components. This file usually need not concern the user unless, for example, new models are being added to the component library.

Key words and user-supplied data are employed to represent the system and to accomplish the various analytic tasks. The 13 primary key words used at present are PROCESS, SYSBEG, SYSEND, VARY, CONSTRAIN, SWEEP, MINIMIZE, PLI, INTEGRATE, STEADY, SWITCH, CONTROL, and DATA. Before considering the SALT language in detail, we present several simple examples.

1.2 EXAMPLES

The following examples treat conventional fossil/steam power plants, but such power systems are by no means the only area of application for SALT. These examples also do not necessarily conform to the requirements of the models presently available for use with SALT; rather, they are intended to give the user some preliminary idea of the language's appearance.

Flow Model for a Simple System

As an initial example, consider a simple steam plant that consists of a steam flow passing through a heater (HT), a steam turbine (ST), a steam condenser (SC), and a water pump (PUMP). Figure 1 depicts the system configuration; in this figure, STM_1 indicates the steam flow, HT_1 the heater, ST_1 the steam turbine, SC_1 the steam condenser, and PUMP_1 the water pump. This configuration is represented within the SALT language by the use of the following PROCESS statement:

```
PROCESS STM_1-> HT_1 ST_1 SC_1 PUMP_1
```

The steam flow is represented by the symbol with the attached arrow, "->"; in general, flow is always represented within the SALT language by a symbol with such an arrow. Each symbol representing a model or a flow consists of a string of characters for the model or flow type, followed by an underscore, "_" and the character "1." This "1" character -- actually a user-defined label that could have been any string of characters -- is used to distinguish between models or flows of the same type, where more than one is used. Thus, if two steam turbines had been used (e.g., for high-pressure and low-pressure stages), they might have been denoted as ST_1 and ST_2 or as ST_HP and ST_LP.

In Fig. 1, the steam flow is shown passing through each of the models; this case is represented within the SALT language by writing the steam-flow symbol before these

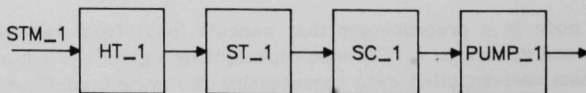


FIGURE 1 Simple Steam Path

models, with the arrow placed after the flow symbol and pointing to the models through which it passes.

Use of the Flow Splitter Model

Figure 2 shows a system configuration in which part of the steam is split off by a splitter model (SP_1), bypasses the heater model and steam-turbine model, and is then remixed with the STM_1 flow in a mixer model (MX_1). The configuration shown in Fig. 2 is represented within the SALT language as follows:

```
PROCESS  STM_1->  SP_1  ->STM_2
          HT_1
          ST_1
          MX_1  <-STM_2
          SC_1
```

The correspondence between the SALT input and the figure remains fairly straightforward. In addition to the pass-through flow used in the first example, this system uses two other flow classes. The first of these is an output flow that originates from a model, such as the STM_2 flow from the SP_1 model. These output flows are, in general, represented by flow symbols written after the models from which they originate, with the arrow pointing to the flow. The second flow class is an input flow, such as the STM_2 flow going into the MX_1 model. This class of flows is again specified by writing the symbol after the model to which it pertains, but with the arrow pointing to the model. Whether a flow is classed as a pass-through, input, or output flow is determined by the relationship of the flow to the model and is not just a property of the flow. Thus, STM_2 above is both an input flow to the MX_1 and an output flow from the SP_1. Input and output flows pertain only to the single model preceding them. In the present example, STM_2 does not interact with the HT_1 or ST_1 models, but STM_1 -- represented as a pass-through flow -- still flows through all of the models.

The actual layout of the SALT input is a matter of individual preference; only the order of the symbols is important. Thus, the SALT input statement corresponding to the system in Fig. 2 could be specified as

```
PROCESS  STM_1->  SP_1  ->STM_2
          HT_1  ST_1  MX_1  <-STM_2
          SC_1
```

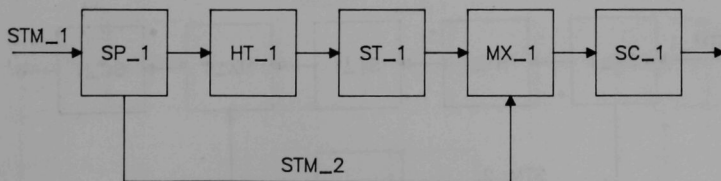


FIGURE 2 Simple Steam Path with Bypass Leg

or even as

```
PROCESS STM_1-> SP_1 ->STM_2 HX_1 ST_1 MX_1 <-STM_2 SC_1
```

The user will find some forms of input statement easier to follow than others.

Use of Parallel System Components

Figure 3 shows a system that uses an additional steam turbine in parallel. This configuration is represented within the SALT language by the following:

```
PROCESS STM_1-> SP_1 ->STM_2
              HT_1 ST_1
STM_2-> ST_2
STM_1-> MX_1 <-STM_2
              SC_1
```

In this example, after the STM_1 flow has been processed through the ST_1 model, STM_2 must be processed through the ST_2 model before both flows can be mixed in the MX_1 model. Thus, the pass-through flow STM_1 is interrupted by simply writing STM_2 as a new pass-through flow through the ST_2 model. The STM_1 flow is then reestablished as a pass-through flow to complete the system configuration. In general, any pass-through flow may be terminated simply by writing another flow as a pass-through flow; the first pass-through flow is temporarily suspended (but not lost or forgotten).

This example also demonstrates that the order of the models is important in representing the system configuration. Also, all the flows must have been processed through the models they pass through before the next model that uses these flows can be specified. Here, it would be incorrect to specify the MX_1 model before the ST_2 model.

Use of the Flow Initiator Model

In the examples presented so far, the incoming STM_1 flow simply passed through the first model within the system configuration. Actually, SALT requires that all flows originate from models (such as the SP model used above). In order to start a flow, a

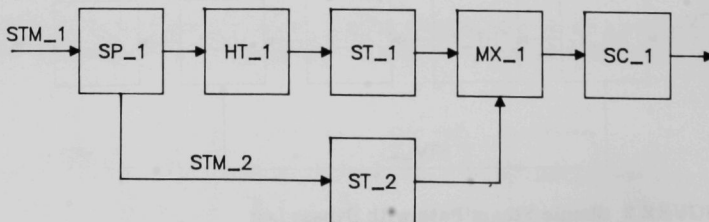


FIGURE 3 Parallel-Path Steam System

special initiator model, denoted IN, is used. This model has only one flow, which technically is an output flow; however, within the SALT input statements it is specified as a pass-through flow. The correct SALT input statement for the first example discussed would actually appear as follows:

```
PROCESS STM_1-> IN_1 HT_1 ST_1 SC_1 PUMP_1
```

The other examples would also use this initiator model. The initiator model assigns initial values to the steam-flow parameters.

Use of Multiple-Entry Models

Figure 4 shows a system configuration that includes a multiple-entry model. This multiple-entry model is a heat exchanger (HX); the hot stream is processed in one entry to the model and the cold stream in another. The two entries are specified at different points within the SALT input. Such entries are specified by adding a colon and an entry designator after the model name. For an HX model, these entry designators are "H" for the hot side and "C" for the cold side. Thus, the system depicted in Fig. 4 is represented in the SALT language as follows:

```
PROCESS GAS_1-> IN_G HX_1:H
      STM_1-> IN_S HX_1:C ST_1 SC_1 PUMP_1 IN_S:CYCL
```

The IN model also is a multiple-entry model. The additional entry to this model, denoted "CYCL," is used to generate the constraints necessary to close the steam loop at the "back door" of the IN S model. The CYCL entry does not by itself close the steam loop; that is done by VARY and CONSTRAIN statements (described below).

Actually, most of the models used by SALT are multiple-entry models. Most models have an "OUT" entry for printing out the results of the calculations. This entry does not require any flows, and a special NULL flow has been provided to terminate any

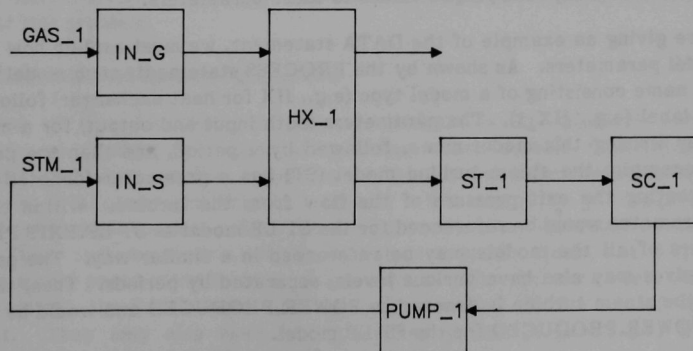


FIGURE 4 System with a Multiple-Entry Model

existing pass-through flow without replacing it with another pass-through flow. For example, to call the output entry to the ST_1 model (where "call" means "specify in a PROCESS statement"), one would write the following statement:

```
PROCESS NULL-> ST_1:OUT
```

(The NULL flow does not require a label.) Of course, all of the model outputs may be specified, as in this version:

```
PROCESS NULL-> IN_G:OUT IN_S:OUT HX_1:OUT ST_1:OUT
               SC_1:OUT PUMP_1:OUT
```

This PROCESS statement (more than one PROCESS statement may be used in the SALT input) should only follow those PROCESS statements that perform the calculations and process the flows. Otherwise, there would be no results to be printed when the model output entries were called. Because calling the output entries is the only way to obtain model output, the SALT code includes an abbreviation for this task:

```
PROCESS NULL-> *_*:OUT
```

Here, " *_*" refers to all of the models called within the system configuration.

Assignment of Values to Model Parameters

So far, we have concentrated on representing simple system configurations by means of PROCESS statements. Actually, even the largest systems can be represented in terms of the same simple methods used with these simple systems. The PROCESS statement itself is the minimal input needed to run SALT. However, the configuration of a system does not by itself represent the system. Each model within the system usually has numerous parameters to which values must be assigned before the system simulation can be performed. (These parameters have default values, but these values may not be the ones required for a specific system.) An additional SALT statement, the DATA statement, is used to assign the proper values to these parameters.

Before giving an example of the DATA statement, we must explain how to reference the model parameters. As shown by the PROCESS statement, each model used in a system has a name consisting of a model type (e.g., HX for heat exchanger) followed by a user-defined label (e.g., HX_1). The parameters (both input and output) for a model are referred to by writing this model name, followed by a period, and then the parameter name. For example, the steam-turbine model (ST) has a parameter (denoted as EXIT_PRES) representing the exit pressure of the flow from the turbine. Within the SALT input, this parameter would be referenced for the ST_LP model as ST_LP.EXIT_PRES. All the parameters of all the models may be referenced in a similar way. The parameter names themselves may also have various levels, separated by periods. Thus, the power produced by the steam turbine is denoted as POWER.PRODUCED and would be referred to as ST_LP.POWER.PRODUCED for the ST_LP model.

7

The DATA statement might then take the following form:

```
DATA  ST_LP.EXIT PRES=0.1;
      ST_LP.EFFICIENCY=0.85;
      PUMP_1.EXIT PRES=120;
      PUMP_1.EFFICIENCY=0.80;
```

Here, the exit pressures and efficiencies of the ST_LP and the PUMP_1 models have been assigned values. Only one DATA statement is allowed, and it must be the last statement within the SALT input; however, the statement may be as long as necessary in order to define values for all of the model parameters. Certain abbreviations are allowed. For example, the model name need not be rewritten before each parameter name; instead, one leaves a blank following the model name:

```
DATA  ST_LP .EXIT_PRES=0.1; .EFFICIENCY=0.85;
      PUMP_1 .EXIT_PRES=120; .EFFICIENCY=0.80;
```

At this point, let us show the complete set of SALT inputs for our first example (see Fig. 1). An initiator model, IN_1, is needed in order to start the steam flow with some values. A typical set of SALT inputs might then take the following form:

```
PROCESS  STM_1-> IN_1 HT_1 ST_1 SC_1 PUMP_1
          NULL->  * *:OUT
DATA      IN_1 .T=600; .P=120; .M=100; .ID='H2O';
          HT_1 .HEAT=1E7;
          ST_1 .EXIT_PRES=1.0; .EFFICIENCY=0.82;
          PUMP_1 .EXIT_PRES=120; .EFFICIENCY=0.80;
```

Here, the steam flow is started at a temperature, T, equal to 600 K; a pressure, P, of 120 atm; and a mass flow rate, M, of 100 kg/s. A total of 10 MW of heat is added to this flow by the HT_1 model; the steam is then expanded to 1 atm at an isentropic efficiency of 82%, condensed at 1 atm, and pumped back to 120 atm at an isentropic efficiency of 80% in the PUMP_1 model. The model output entries are then called to print out the results of the simulation. This PROCESS statement and DATA statement are all that are needed for this problem.

Calculation of System Parameters

One of the functions of systems analysis is to calculate certain system parameters, as opposed to model parameters. Such system parameters are often functionally dependent on the collective values of the model parameters. Thus, for example, the total power produced by a system is the sum of the powers produced by the individual component models. Such system parameters are obtained by calling system models. These systems models usually do not process any flows, but they can process individual model parameters in each of the models. The system models themselves are specified within the SALT input (like any other model) by the use of the PROCESS statement. They may also have multiple entry points; in particular, there may be multiple output entries for the printing not only of the system parameters, but also (possibly) of tabular summaries of the individual model parameters used in calculating such system parameters.

For example, the SYST system model is used to calculate power summaries and to print such summaries when its OUT entry is specified. This model will also print out tables of the exit flow conditions from each model, by flow name. A concise overview of the system simulation can be obtained by calling only this one output entry, rather than calling all of the individual model outputs. In the complete SALT input for the first example, the SYST and SYST:OUT entries could be called simply by replacing the one line

```
NULL-> * _*:OUT
```

with the line

```
NULL-> SYST_1 * _*:OUT
```

The system model, like all models, requires the user-defined label; the "*" notation includes any system models. Of course, any input parameters that the SYST_1 model requires must also be added to the DATA statement.

Using the PROCESS and DATA statements, it is possible to model many systems; however, as indicated above, closed flow loops actually require the use of additional SALT language statements. The real power of the SALT code lies in its use of these additional statements to establish system constraints, perform optimizations or parametric studies, etc.

Use of System Constraints

Consider the first system (shown in Fig. 1) again. Suppose we wish to make the total power produced by the ST_1 model equal to ten times the power consumed by the PUMP_1 model and that the exit pressure of the ST_1 model is to be varied to establish this constraint. This type of problem is symbolized using the VARY and CONSTRAIN statements, as follows:

```
VARY ST_1.EXIT.PRES = 1.0 0.01 5.0
CONSTRAIN ST_1.POWER.PRODUCED = 10*PUMP_1.POWER.CONSUMED
```

The three numbers within the VARY statement represent an initial value for the steam-turbine exit pressure and lower and upper bounds between which the exit pressure may be varied. The actual values that the exit pressure takes in order to establish the constraint are dictated by an equation solver. This equation solver uses an iterative technique, starting with the initial value furnished in the VARY statement.

In many cases, constraints can be thought of as constraining some subsystem, rather than the entire system. These subsystems are delimited, within the SALT inputs, by means of SYSBEG and SYSEND statements. These statements, consisting of the SYSBEG and SYSEND key words followed by a delimiting user-defined label, are interspersed among the other SALT language statements to define the beginning and ending points of a subsystem.

VARY and CONSTRAIN statements, as well as other SALT language statements that define iterative tasks, must always be included between the SYSBEG and SYSEND statements. Thus, the SALT input for our simple system (Fig. 1) -- with the constraint included -- might appear as follows:

```

SYSBEG A
  PROCESS STM_1-> IN_1 HT_1 ST_1 SC_1 PUMP_1
  VARY ST_1.EXIT PRES = 1.0 0.01 5.0
  CONSTRAIN ST_1.POWER.PRODUCED=10*PUMP_1.POWER.CONSUMED
SYSEND A
PROCESS NULL-> SYST_1 * *:OUT
DATA .
.
.

```

The SYSBEG-SYSEND delimiter label used is "A," and the subsystem actually taken is the entire system configuration. In this case, varying the steam-turbine exit pressure does not affect the models upstream, so the same results would be obtained by defining the subsystem only around the ST_1, SC_1, and PUMP_1 models; then the IN_1 and HT_1 models would be called only once before the iterations within the subsystem were performed. The overhead of calling the IN_1 and HT_1 models many times would be avoided, and a faster and less expensive computer run would result. With the subsystem only around the ST_1, SC_1, and PUMP_1 models, the SALT inputs look like this:

```

PROCESS STM_1-> IN_1 HT_1
SYSBEG A
  PROCESS STM_1-> ST_1 SC_1 PUMP_1
  VARY ST_1.EXIT PRES = 1.0 0.01 5.0
  CONSTRAIN ST_1.POWER.PRODUCED=10*PUMP_1.POWER.CONSUMED
SYSEND A
PROCESS NULL-> SYST_1 * *:OUT
DATA .
.
.

```

Any number of VARY and CONSTRAIN statements can be included within a subsystem. For example, in addition to the constraint specified above, suppose it were required to constrain the outlet temperature of the heater, denoted as HT_1.TEMP, to 800 K (for a fixed heat load) by varying the steam-flow rate between 100 and 200 kg/s. This could be stated as follows:

```

SYSBEG A
  VARY IN_1.M = 150 100 200
  CONSTRAIN HT_1.TEMP = 800
  VARY ST_1.EXIT PRES = 1.0 0.01 5.0
  CONSTRAIN STM_1.POWER.PRODUCED=10*PUMP_1.POWER.CONSUMED
  PROCESS STM_1-> IN_1 HT_1 ST_1 SC_1 PUMP_1
SYSEND A
PROCESS NULL-> SYST_1 * *:OUT
DATA .
.
.

```

The VARY and CONSTRAIN statements need not always be written at the end of the subsystem; they can be written between any statements within the subsystem, and their order is immaterial.

For this particular problem, the constraints and the parameters varied to establish them can actually be split into two subsystems, as follows:

```

SYSBEG A
  VARY IN 1.M = 150 100 200
  CONSTRAIN HT 1.TEMP = 800
  PROCESS STM_1-> IN_1 HT_1
SYSBEG A
SYSBEG B
  PROCESS STM 1-> ST 1 SC 1 PUMP 1
  VARY ST 1.EXIT PRES = 1.0 0.01 5.0
  CONSTRAIN ST 1.POWER.PRODUCED = 10*PUMP_1.POWER.CONSUMED
SYSBEG B
PROCESS NULL-> SYST_1 *_*:OUT
DATA .
.
.

```

This form is slightly more efficient computationally than the previous form, but both approaches should yield the same results within numerical accuracy. The single-subsystem form attacks the problem as two equations in two unknowns; the two-subsystem form, as two sets of one equation in one unknown. It is also possible to have subsystems nested within other subsystems.

Optimization of System Parameters

The constraint specified above concerning the steam-turbine power produced is somewhat contrived. A more realistic goal might be to maximize the turbine power produced.* This type of problem, an optimization, is treated in the SALT language by using the MINIMIZE statement. (By minimizing the negative of an expression, one obtains the maximum of that expression.) Thus, maximizing the steam power produced is represented as follows:

```
MINIMIZE -ST_1.POWER.PRODUCED
```

This statement would replace the constraint used previously, as follows:

```

PROCESS STM_1-> IN_1 HT_1
SYSBEG A
  VARY ST 1.EXIT PRES = 1.0 0.01 5.0
  MINIMIZE -ST 1.POWER.PRODUCED
PROCESS STM_1-> ST 1

```

*Net power produced by the system would be an even better objective function. The example is only used to illustrate the SALT language.

```

SYSEND  A
PROCESS STM_1-> SC_1 PUMP_1
        NULL-> SYST_1 *_*:OUT
DATA    .
        .
        .

```

Both the parameter being varied and the objective function of the MINIMIZE statement pertain only to the ST_1 model. Therefore, this model is the only one included in the subsystem.

Essentially, the system in question has one degree of freedom; one variable is varied in order to maximize the steam-turbine power. For an optimization problem, additional variables may also be varied, leading to systems having higher degrees of freedom. Suppose that in addition to the turbine exit pressure, the heat transferred in the HT_1 model and the inlet steam-flow rate were also varied, as represented by the following statements:

```

SYSBEG  A
PROCESS STM_1-> IN_1 HT_1 ST_1
VARY    IN_1.M_ = 150 100 200
VARY    HT_1.HEAT = 1E6 1E5 1E7
VARY    ST_1.EXIT PRES = 1 0.01 5.0
MINIMIZE -ST_1.POWER.PRODUCED
SYSEND  A
PROCESS STM_1-> SC_1 PUMP_1
        NULL-> SYST_1 *_*:OUT
DATA    .
        .
        .

```

If additional parameters pertaining to additional models are varied, these models also must be included in the subsystem.

It is possible to include the CONSTRAIN statement within optimization tasks. Thus, to constrain the HT_1 model exit temperature to a value of 800 K, one simply adds the the statement:

```
CONSTRAIN HT_1.TEMP = 800
```

to the subsystem. By adding such an equality constraint, the degrees of freedom are reduced from three to two. If two more equality constraints were added, then the problem would have zero degrees of freedom; the parameters being varied would be just sufficient to satisfy the constraints, with none remaining for the minimization. It is possible to include more constraints than parameters being varied; however, such constraints must be inequalities. For example, if it were enough to keep the exit heater temperature greater than 800 K in the foregoing example, then the constraint would be written as follows:

```
CONSTRAIN HT_1.TEMP > 800
```

This constraint would not necessarily reduce the degrees of freedom from three to two.

As with subsystem tasks handled by pure-equality constraints (i.e., nonoptimization tasks), more than one optimization subsystem task may be used, and the tasks may also be nested. Optimization tasks may also be used along with nonoptimization tasks. Numerous ways may exist in which to set up the same basic problem using the SALT language. Of course, any problems set up should be well posed.

Use of the Parameter Sweep

Instead of being interested in the single value of some parameter that solves an optimization problem or establishes a constraint, one might wish to consider the results of the system simulation for an entire range of parameter values (sensitivity analysis). Such parameter-sweep problems are defined using the SWEEP statement. For example, to see the effects of varying the ST_1.EXIT_PRES from 1 to 5 atm in increments of 0.5 atm, one would write:

```
SWEEP ST_1.EXIT_PRES = 1 TO 5 BY 0.5
```

Sweeping tasks, being defined within a subsystem, require the SYSBEG and SYSEND subsystem delimiters. Taking our simple example of Fig. 1 and including this sweeping task, we have the following for the SALT inputs:

```
SYSBEG A
  SWEEP ST_1.EXIT_PRES = 1 TO 5 BY 0.5
  PROCESS STM 1-> IN 1 HT 1 ST_1 SC_1 PUMP_1
    NULL-> SYST_1 **:OUT
SYSEND A
DATA .
.
.
```

Unlike the other tasks, this sweeping task also includes the calls to the model output entries, as well as the SYST_1 model, within the task. If the entries were called within the task rather than after the SYSEND statement, then only the results for the last turbine exit pressure would be printed, defeating the purpose of the sweep. Sweeping tasks should not include VARY, CONSTRAIN, or MINIMIZE statements; however, additional subsystems that do include these statements may be nested within the sweeping tasks.

1.3 JOB-CONTROL LANGUAGE

The sequence of steps required in running the SALT system code after the STRUCT file has been prepared is included in a job-control-language (JCL) procedure used on the IBM computer system at Argonne National Laboratory (ANL). (The SALT code itself exists on computer tape at the National Energy Software Center, located at ANL.) This JCL procedure is reproduced in App. A.

2 SALT LANGUAGE

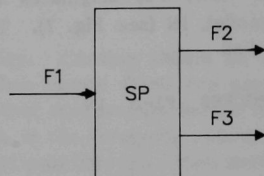
This chapter describes the elements of the SALT language. The SALT code currently uses 13 different primary key words to represent the system and the various tasks that are to be performed (see App. B). Each of these key words and the statements associated with them are discussed in the following sections.

2.1 PROCESS STATEMENT

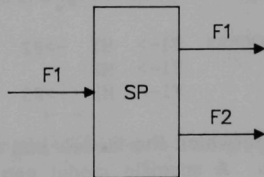
The key word **PROCESS** begins a **PROCESS** statement, which defines the components of the system and indicates how such components are connected by flows. The system components usually represent such devices as pumps, compressors, heat exchangers, turbines, etc., but they may also represent purely computational procedures to be performed within the system analysis. Similarly, the flows of the system usually represent the flows of gases, steam, air, coal slurries, etc., but they also may represent simply the flow of information from component to component.

Each component within the component library will process either no flows at all or one or more flows of a specific type. These flows can be categorized as pass-through, input, or output flows. A pass-through flow is one that both enters and leaves a given component. A flow that enters but does not leave a component is an input flow, and a flow that leaves but does not enter a component is an output flow. The specification of flows as pass-through, input, and output flows is dictated by the developer of the model rather than by the user of the system code. Thus, for example, a splitter model (SP) might have been written to handle three flows -- one input flow, F1, and two output flows, F2 and F3 (see Fig. 5a). Alternatively, the SP model might have been written to process only two flows -- one pass-through flow, F1, and one output flow, F2 (see Fig. 5b).

In using the **PROCESS** statement to describe a system configuration, one specifies the pass-through flows first, followed by the model name, and then the input and output flows. If a model has no pass-through flow, then at least one input flow should be specified before the model name.



(a) One Input Flow and Two Output Flows



(b) One Pass-Through Flow and One Output Flow

FIGURE 5 Alternative Flow Arrangements for the Splitter Model

Consider the system shown in Fig. 6. It consists of only one component, M1, which processes one pass-through flow, F1; one input flow, F2; and one output flow, F3. This is represented in the SALT code by the following statement:

```
PROCESS F1-> M1 <-F2 ->F3
```

The symbols with the arrows represent the flows; those without arrows (except the word PROCESS) represent the models. Pass-through flows are written before the model, input and output flows after the model. Pass-through and input flows have the arrow pointing to the model, while output flows have the arrow pointing to the flow. The only exception to this rule occurs when a model has no pass-through flow; in such a case, the first input or output flow to the component should be written as if it were a pass-through flow. For example, suppose a flow, F1, originates from an initiator model, IN (see Fig. 7). The configuration shown in Fig. 7 would be correctly symbolized as

```
PROCESS F1-> IN
```

but not as

```
PROCESS IN ->F1
```

For systems with many components, the PROCESS statements for each individual component are simply strung together without specifying the keyword PROCESS again. Thus, the system shown in Fig. 8 is symbolized as follows:

```
PROCESS F1-> M1 ->F2
      F1-> M2
      F1-> M3 ->F3
```

The order in which the models and their flows are written down depends on which flows are known. A specific model can be written down only when all the other models generating all the input or pass-through flows for that model have been previously specified.

The SALT code will "remember" previous pass-through flows, so such flows need not be specified each time a component is specified; this feature serves to simplify the PROCESS statement. Thus, if a model is written down without a pass-through flow, it is

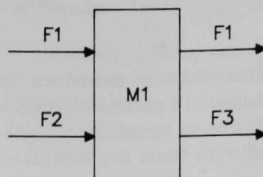


FIGURE 6 Model with One Pass-Through Flow, One Input Flow, and One Output Flow

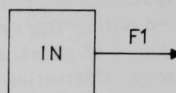


FIGURE 7 Flow-Initiator Model

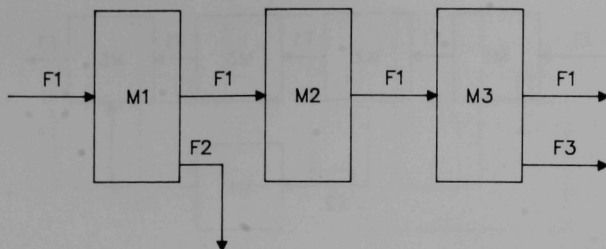


FIGURE 8 Simple Three-Model System

assumed that the last-occurring pass-through flow also goes through this model. The example shown in Fig. 8 can be written as

```

PROCESS  F1->  M1  ->F2
          M2
          M3  ->F3
  
```

or (because there is no need to skip to the next line for each model) as

```

PROCESS  F1->  M1  ->F2  M2  M3  ->F3
  
```

The user may arrange the layout for maximum clarity. Caution should be used in specifying many components and flows on one line when output flows are generated, because these output flows do not pass through the other components unless they are written explicitly as pass-through or input flows. Thus, in the foregoing example, flow F2 does not go through models M2 or M3. Models without pass-through flows, such as the flow-initiator model, have at least one flow written as a pass-through flow, because otherwise the previously occurring pass-through flows would be passed to such a model.

The processing of a flow may be interrupted for the processing of other flows and resumed later. Consider the example illustrated in Fig. 9. This configuration would be represented as follows:

```

PROCESS  F1->  M1  M2  ->F2
          F2->  M4
          F1->  M3  M5  <-F2
  
```

The processing of F1 is interrupted after M2, in order that F2 can be obtained at the exit of component M4 before M5 is called.

In writing down the model name and flows, the order of the flows with respect to the model is important. If a model has two pass-through flows, the user must be aware of which flow is to be specified first. This requirement also extends to the input and output flows. For example, suppose a model (M1) requires two pass-through flows and

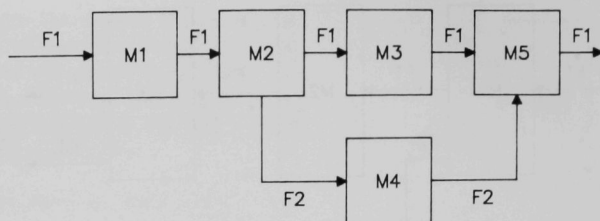


FIGURE 9 Simple Five-Model System

one output flow (see Fig. 10). Then F1 and F2 must be in the correct order when the PROCESS statement is written. The statement

```
PROCESS F1-> F2-> M1 ->F3
```

is not the same as

```
PROCESS F2-> F1-> M1 ->F3
```

(The first of these two statements is the correct one.) The ordering of the flows

with respect to the model would be decided upon by the developer of the model. All of the flows required by any model must be specified within the SALT input. For example, suppose a steam-turbine model is developed that includes a feedwater-extraction flow as an additional output; that flow must be specified within the SALT input, even if the flow is assigned a zero mass-flow rate (i.e., the flow is not used within the system).

The SALT code also makes it possible for a model to process multiple flows; either one flow or several flows can be processed at a time. For example, the heat-exchanger model was written to process the hot flow in one component call and the cold flow in another call. In order for SALT to identify which flow such a component is processing, an optional entry label is added to the model name by appending a colon, followed by the entry name. In the case of the HX model, these entry names consist of "H" for the hot flow and "C" for the cold flow. In the system diagram below (Fig. 11), a flow (F1) originates from the IN model, passes through the hot side of a heat exchanger (HX), is further processed by another model (M1), and finally is fed back through the cold side of the heat exchanger. The SALT representation of this diagram would be as follows:

```
PROCESS F1-> IN HX:H M1 HX:C
```

If no entry label is used, the SALT code assumes the entry label "C." In some cases, models have been written such that the processing of the first flow must be done before

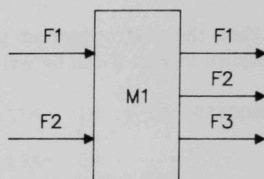


FIGURE 10 Model with Multiple Pass-Through Flows

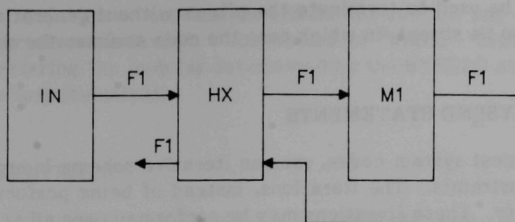


FIGURE 11 Example Using a Multiple-Entry Model

that of the other flows of the model. An example of this is the MHD channel model, MG, where the first flow is the hot-gas flow and the second flow is the cooling-water flow. The MG model calculates the heat removed from the hot-gas flow when processing the first flow. The heat removed is retained within the model and is used when the model processes the cooling-water flow.

In the foregoing examples, the fictitious flows and models have been named F1, F2, M1, M2, etc. In an actual system problem, the names should correspond to the names of the flows and models available in the component library and defined within the INTF file. Each model or flow used in the PROCESS statement takes the form of a model type (e.g., HX for heat exchanger) or a flow type (e.g., GAS for gas flow), followed by an underscore and a user-defined label. This label, which can be up to ten characters in length, is used to delineate multiple components of the same type. Thus, if a system configuration requires the use of two heat-exchanger models, these models could be denoted as HX_1 and HX_2 or HX_SH and HX_RH (for a superheater and a reheater). These optional labels are attached directly to the model-type name without any separating blanks. For the sake of simplicity, we will usually set this label to "1" in our examples.

The general form of the process statement can now be summarized as follows:

```
PROCESS spec spec spec . . .
```

where spec takes the form

```
pflow-> ... model:entry <-iflow ... ->oflow ...
```

Here, the use of ellipsis indicates that the preceding symbol may appear more than once. "Pflow," "iflow," and "oflow" stand for the pass-through, input, and output flows, respectively, each of which takes the form of a flow type concatenated with an underscore and a user-defined label of ten characters or fewer. "Model," which represents the model name, takes the form of a model type concatenated with an underscore and a user-defined label, also of ten characters or fewer. "Entry" represents the optimal entry label. Any or all of the flows may be absent. If no pflows appear, then the last-appearing pflows are assumed. The special pflow denoted "NULL->" and used

without a label can be used to terminate the pflows without generating a new one. The model entry may also be absent, in which case the code assumes the entry "C."

2.2 SYSBEG AND SYSEND STATEMENTS

SALT, like most system codes, uses an iterative scheme in order to meet various types of system constraints. The iterations, instead of being performed automatically, are set up by the user. These iterations may be performed over all or part of the system configuration. The SYSBEG and SYSEND statements delimit the beginning and end of an iterative loop. Each of these key words is followed by the same user-defined label, which must be different for each iterative loop defined; The form of each of these statements is as follows:

```
SYSBEG  label
SYSEND  label
```

These key words may also define nested loops. The only requirement is that each loop be either fully contained in or fully excluded from other loops; partial overlapping is not permitted.

The actual function performed within the SYSBEG-SYSEND loop is dictated by VARY, SWEEP, or INTEGRATE statements. At least one such statement must be specified in each SYSBEG-SYSEND loop, with only one of these types specified in any one loop.

2.3 VARY STATEMENT

The VARY statement defines the variables to be varied in order to meet specified constraints or perform an optimization. The form of the VARY statement is as follows:

```
VARY  variable_name = start lower upper
      ; variable_name = start lower upper
      .
      .
      .
```

Here, "variable name" is the name of the variable to be varied, "start" is the starting value, and "lower" and "upper" specify lower and upper bounds on the variable. The starting value should be specified between these lower and upper bounds. If "start" is given the value "*", the variable's current value will be used as the starting value. Any number of variables can be specified, but each must be separated from the others by a semicolon; this semicolon is a separator, not a terminator, so the last upper bound should not be followed by a semicolon.

In the VARY statement, the starting value and the lower and upper values may be either variables or algebraic expressions. In any case, the character string

representing these variables or expressions should not contain any blanks and should be fewer than 31 characters in length. The values of "start," "lower," and "upper" are evaluated before entering the loop (as determined by the SYSBEG statement) and are not changed during the loop iterations.

2.4 CONSTRAIN STATEMENT

The statement used to specify constraints takes the form

```
CONSTRAIN  exp op  exp
           ;  exp op  exp
           .
           .
           .
```

where "exp" is an algebraic expression of variables known at the end of the loop in which the constraint statement occurs and "op" is a relational operator ("=", "<," or ">").

The use of inequalities in a CONSTRAIN statement is permitted only when the MINIMIZE statement is used to define an optimization problem. Algebraic expressions used in CONSTRAIN statements should be fewer than 72 characters in length. If longer expressions are needed, they can be shortened to single variables by means of the PLI statement (see Sec. 2.12), and the single variables can then be used in the CONSTRAIN statement. As was the case with the VARY statement, the semicolon is used to separate constraints but not to terminate them.

When no MINIMIZE statement has been specified, the number of equality constraints must equal the number of variables within the VARY statements for the SYSBEG-SYSEND loop.

2.5 MINIMIZE STATEMENT

The minimize statement, which defines objective functions for optimizations, takes the following form:

```
MINIMIZE  exp
```

Here, "exp" is an algebraic expression of variables known at the end of the loop in which the statement occurs. As with the expressions used in the CONSTRAIN statement, there is a length restriction of 72 characters. Longer expressions may be reduced by using PLI statements. If MINIMIZE is used within a loop, the number of variables in the VARY statement should be greater than the number of equality constraints; otherwise, there will be no extra degrees of freedom available over which to perform the optimization.

Since $\max f(x) = -\min(-f(x))$, one can maximize a function by minimizing the negative of that function.

2.6 SWEEP STATEMENT

Used to define a parameter sweep, the SWEEP statement takes the following form:

```
SWEEP  variable_name = specification
      ; variable_name = specification
      .
      .
      .
```

Here, "variable_name" is the name of the variable to be swept over and "specification" is any legitimate PL/I do-loop specification. Thus, the following are all valid SWEEP statements:

```
SWEEP  variable_name = value1 TO value2 BY value3
SWEEP  variable_name = value1, value2, value3 TO value4
SWEEP  variable_name = value1, value2 WHILE( exp1 = exp2 )
```

The last example shows that conditional termination of the sweep is also possible. In this case, the variable being swept over takes the values "value1" and then "value2," but only if "exp1" = "exp2." The PL/I do-loop UNTIL option may also be used in the SWEEP statement.

The specifications are limited to 72 characters, including the variable name and "=" sign. If the SWEEP statement is used within a SYSBEG-SYSEND loop, then CONSTRAIN, MINIMIZE, or VARY statements should not be used within this same loop; however, these types of statements may be used in another SYSBEG-SYSEND loop nested within the SWEEP loop.

2.7 INTEGRATE STATEMENT

The INTEGRATE statement defines the starting time and output times used within the dynamic simulation. The last specified output time also defines the termination time. The form of the INTEGRATE statement is as follows:

```
INTEGRATE  TSTART=tstart TOUT= spec , spec , spec , spec . . .
          METH=value TASK=value RTOL=value ATOL=value
```

where "tstart" is the starting time (usually zero) unless the current job is a restart from a previous computer run. "Spec" takes one of the following two forms:

```
value TO value BY value
```

or

```
value
```

The first TOUT value should be a number greater than the TSTART time. Output is always generated for the TSTART time. METH is set either at 10 (for an explicit variable-order, variable-step Adams method up to twelfth order) or 21 (for Gear's stiff method). TASK is used to prevent integrating beyond the TOUT values if set to a number greater than 1. TASK = 1 is the normal mode, in which the integrator may integrate beyond the TOUT values and then interpolate at the output values. RTOL and ATOL are relative and absolute tolerances used within the integration. If the parameters TSTART, METH, TASK, RTOL, and ATOL are omitted, they assume the values 0.0, 21, 1, 10^{-4} , and 10^{-4} , respectively.

2.8 STEADY STATEMENT

The STEADY statement is used to generate a steady-state starting point for the dynamic simulation. The form of the steady statement is as follows:

```
STEADY RTOL=value ATOL=value
```

where RTOL and ATOL are optional parameters specifying the relative and absolute tolerances used in determining the steady-state solution.

2.9 CONTROL STATEMENT

The CONTROL statement defines dynamic-system controls. These may take the form of additional equations to be integrated or algebraic equations to be solved at each value of time. The general form of the CONTROL statement is as follows:

```
CONTROL spec ; spec ; spec . . .
```

Here, "spec" is expressed either as

```
D_DT variable = expression
```

or as

```
expression1=expression2 USING variable
```

The first form defines an additional differential equation, where D_DT represents the time-derivative operator. The second form is used to define algebraic constraints, where "expression1" is constrained to equal "expression2" by suitably varying the "variable" specified after the key word USING.

2.10 DATA STATEMENT

The DATA statement defines the values of the various input parameters used in the models. Where it is used, the DATA statement should be the last statement within the STRUCT file. The general form of the DATA statement is as follows:

```
DATA header variable = value;
    header variable = value;
    .
    .
    .
```

where "header" is optional and is used to store major levels of PL/I aggregate variables to be used with succeeding variables. If "header" is used, it should be followed by a blank space, and all variables belonging to this PL/I aggregate should begin with a period.

As an example, suppose we wish to assign values to two parameters (say, PARM1 and PARM2) of a model M1 and to three parameters (PARM1, PARM2, and PARM3) of a model M2. We would accomplish this by the following statements:

```
DATA M1 .PARM1=value; .PARM2=value;
      M2 .PARM1=value; .PARM2=value; .PARM3=value;
```

Other input parameters for other models may be assigned values in like fashion. No requirements govern the order of parameter assignment. (The names and meanings of all the model parameters are presented in Chapter 4.)

It is also possible to use algebraic expressions in place of "value." In this case, any variable name used should have a value before the expression is encountered. For example, continuing with the previous DATA statement, one could write:

```
M3 .PARM1=M1.PARM1*M2.PARM2;
```

2.11 SWITCH STATEMENT

The SWITCH statement is used to define various controlling parameters whenever the equation solvers or optimizers are used (i.e., whenever VARY has been specified). The form of the statement is as follows:

```
SWITCH MAXIT=value DEL=value ACC=value PRINT=value
```

where "value" is a number representing the value of the preceding parameter, MAXIT is the maximum allowable number of iterations that may be performed within this loop, ACC is the termination criterion, DEL is a value used to determine perturbations of the independent variables, and PRINT is a print switch used to obtain output from the equation solvers and optimizers.

2.12 PLI STATEMENT

The PLI statement is used to code PL/I statements within the STRUCT file. The PLI key word should appear before any such statements are listed. All PL/I statements should be terminated by a semicolon. SALT stores PL/I statements internally as character strings of finite length; no PL/I statements should contain strings of nonblank characters longer than about 50 characters. Any other SALT key word will terminate the insertion of PL/I statements, so the use of SALT key words within PL/I comments should be avoided.

3 MODELS

3.1 MODEL AND FLOW VARIABLES

In general, each model within the system has associated with it a PL/I structure variable containing not only the input parameters of the model, but also all of the output parameters and saved values of the flows processed by the model. The flows are also defined by PL/I structure variables containing the various parameters associated with the flows.

It is not entirely necessary to know the structure of the flow variables in order to use the SALT code, but it is useful to have some idea of what variables are carried along with the flow. The gas flow (GAS), a typical flow variable, is defined by the following PL/I structure:

```

1 GAS,
  2 NAME CHAR(16),
  2 ID CHAR(4),
  2 ATOM(8) FLOAT(16),
  2 PROP,
    3 (TEMP,PRES,ENTH,ENTP,QUAL,RHO,VEL,MASS) FLOAT(16),
  2 COMP,
    3 (XAR,XCH4,XCO,XCO2,XH,XH2,XH2O,XH2S,XK,XKOH,XNO,XN2,
      XO,XOH,XO2,XSO2,XHCL,XCH3OH,XC,XCOS,XNH3,XS,XCL) FLOAT(16),
  2 SOL,
    3 WTF FLOAT(16),

```

In this structure, NAME is a character string containing the name of the flow as used within the SALT input. The ID character string is used to define the type of properties code used with the flow. The substructure PROP contains the variables defining the thermodynamic conditions of the flow. The individual elements of PROP -- TEMP, PRES, ENTH, ENTP, QUAL, RHO, VEL, and MASS -- are the flow's temperature, pressure, enthalpy, entropy, quality, density, velocity, and mass flow rate, respectively.* ATOM(8) represents the atomic weight fractions of the flow constituents -- argon, carbon, hydrogen, potassium, nitrogen, oxygen, sulfur, and chlorine. The substructure COMP is used to hold the molar fractions of the individual species in the flow. Thus, XAR represents the molar fraction of Ar, XCH4 represents the molar fraction of CH₄, and so on. The solids weight fraction entrained in the flow is represented by SOL.WTF.

Not all of the variables listed here are used with all flows. Thus, if the value assigned to ID is "GAS," then all variables within the flow are used; however, if ID is "H₂O," only the information within the PROP substructure is used.

The form of the flow structure is not dictated by the SALT code but by the developer of the models. In general, SALT can handle any type of flow structure.

*In general, SI units are used for all quantities associated with SALT models; pressure, however, is measured in atmospheres. See App. C for further details.

As an example of a typical model PL/I structure, the gas-turbine (GT) model structure takes the form:

```

1 GT BASED(GT P),
  2 NAME CHAR(16),
  2 FLC,
    3 FNAME CHAR(16),
    3 ID CHAR(4),
    3 ATOM(8) FLOAT(16),
    3 PROP,
      4 (TEMP,PRES,ENTH,ENTP,QUAL,RHO,VEL,MASS) FLOAT(16),
    3 COMP,
      4 (XAR,XCH4,XCO,XCO2,XH,XH2,XH2O,XH2S,XK,XKOH,XNO,XN2,
        XO,XOH,XO2,XSO2,XHCL,XCH3OH,XC,XCOS,XNH3,XS,XCL) FLOAT(16),
    3 SOL,
      4 WTF FLOAT(16),
  2 PARM,
    3 DDNAME CHAR(7),
    3 MODE CHAR(10),
    3 EFFICIENCY FLOAT(16),
    3 MECH EFF FLOAT(16),
    3 EXIT_PRES FLOAT(16),
    3 MASS_FACT FLOAT(16),
    3 M_FACT FLOAT(16),
    3 PRDES FLOAT(16),
    3 PRES_RATIO FLOAT(16),
    3 INIT_BIT(1),
  2 POWER,
    3 (INPUT,PRODUCED,CONSUMED,LOSS) FLOAT(16),
  2 PRATIO,
    3 PTR POINTER,
  2 COST FLOAT(16);

```

Each variable within such a structure is referenced by adjoining the higher-level structure names. Thus, the exit pressure of the gas turbine, denoted as EXIT_PRES, is designated by GT.PARM.EXIT_PRES. (Intermediate-level names may be omitted if the resulting name is unambiguous.)

In general, the PARM substructure of any model represents all of the inputs to the model (excluding the flows). The PARM substructure is different for each model. In addition to the input parameters, the PARM substructure may also contain various output parameters that are calculated by the model. Other output parameters may be put into other substructures, which may be of similar form for all models. For example, the POWER substructures of all models are of the same form; this common form permits these substructures to be operated on in the same way for each model, as in summing up the POWER structures to determine plant power.

The flows processed by a model represent some of the most important outputs from the model, so most of the flows (on leaving the model) are saved in model substructures. These substructures, at least for the flows structurally similar to GAS, are denoted as "FL" followed by the entry name used to process the flow; if the entry processes more than one flow, an additional designator follows the entry name. For

instance, the heat-exchanger model (HX) has two flow substructures, saving the hot and cold flow streams from the model. The hot stream, processed by the "H" entry to the model, is saved in the substructure named "FLH;" the cold flow, processed by the "C" entry, is saved in "FLC." In this case, because each entry only processes one flow, no additional designations after the entry name are needed. The feedwater-heater model (FH) also has a hot ("H") and a cold ("C") entry point. The hot entry processes two flows, which are saved in the substructures FLH1 and FLH2. After a model has performed its calculations, the exit flow conditions (temperature, pressure, etc.) can always be referred to using these variables (FLH, FLC, etc.). Thus, the value of the exit temperature from GT_1 is contained in GT_1.FLC.TEMP.

Before any model is called (i.e., specified in a PROCESS statement), each of its input variables must have a value assigned to it. These assignments may be done in three different ways. The first way is to use the initial attribute, giving the variables a default value; this is accomplished within the INTF file and is described later. (Most of the input variables have some input default value.) The second way of assigning input values is to use one of the SALT language statements (e.g., VARY, SWEEP, or PLI assignment statements). Thus, one could write

```
PLI ST_1.PARM.EXIT_PRES=10;
```

to assign a value of 10 atm to the gas-turbine exit pressure. The third way is to use the DATA statement, as in the following:

```
DATA ST_1.PARM.EXIT_PRES=10.;
```

3.2 INITIALIZATION OF PROPERTIES CALCULATIONS

Most of the models process one or more fluid flows and are thereby required to perform calculations of thermodynamic or transport properties. Although detailed knowledge of these property-calculation procedures is not necessary, the user of the SALT code does need to know how to initialize these procedures. This is done by calling the "IN" entry in the general properties model (GP), using the following PROCESS statement:

```
PROCESS GP_1:IN
```

This entry does not require any flows and should be specified before any other model is called. It will read one or more data files, depending on the options specified in the GP model's parameters, so it should not usually be included within any iterative tasks.

At present, steam and water properties are calculated by a procedure similar to that of the WASP code from the NASA-Lewis research center. Additionally, a very fast cubic B-spline fit of the steam properties is available for use in dynamic-system runs. Combustion-gas properties and equilibrium compositions are calculated by minimizing the Gibbs free-energy function. The GAS properties code, at present, handles only 23 chemical species - Ar, CH₄, CO, CO₂, H, H₂, H₂O, H₂S, K, KOH, NO, N₂, O, OH, O₂, SO₂, CH₃OH, HCl, C, COS, NH₃, S, and Cl. A generic condensable-pure-substance

properties code is also available that can individually handle more than 400 different chemical species (chiefly light hydrocarbons). This procedure is based on the Lee-Kesler equation of state. Some special procedures are provided for handling liquid-metal properties. (Not all properties codes are, at present, available for use with all models.) All of the codes have been designed to interact with the component models by means of a common calling sequence (i.e., by calling GP) in order to make the future expansion of the code easier.

The actual properties procedure used by any flow is determined by the flow's ID. This ID value is one of the parameters of the flow (provided the flow is of the GAS type) and is initially assigned a value, like all the parameters of the flow, using the flow initiator model (see Sec. 4.2.7). If ID is assigned the value "GAS," the gas properties code is used with the flow. If ID is assigned the value "H2O," then the steam-water procedure is used. If ID equals "THR," "JAN," or "LIQ," then the condensible-fluids code, the special code for handling single-species gases, or the special single-phase liquid code, respectively, is called. These last three codes may handle more than one type of fluid, so the name of the fluid is added to the ID string after "THR," "JAN," or "LIQ." For example, to use the liquid code for sodium, the ID would be "LIQ NA." The actual fluid names that can be used with each code are listed in the appendix. (A chemical-equilibrium property routine for solutions is also a part of SALT, but the routine is not available in the present version of the code.)

3.3 DEMAND-TYPE MODELS

Some of the models developed require certain conditions to be met within the model that really put demands not on the input parameter of that model, but rather on input flow conditions beyond the model's control. The steam-drum model, for instance, is designed to work only with an input flow of a specified quality. The steam-drum model itself cannot simply alter its input flow, because the flow originates somewhere upstream of the drum.

Demand-type models are developed whenever a real physical device actually creates or changes conditions upstream of itself. They may also be developed to improve computational robustness or efficiency when analyzing the system; the demand-type steam drum was, in fact, developed for this latter reason. The demand-type steam-drum model generates two saturated flows -- a liquid and a steam flow -- for whatever input flow exists, even if it is subcooled. In this way, there will always be a nonzero steam flow for use in any turbine train. The constraint on the inlet enthalpy to the drum, necessary to provide these two flows, is established using the SALT language.

Some demand-type models exist only in certain modes of operation. For example, the heat-exchanger model, in the off-design mode, is a demand-type model. The model calculates a heat-transfer surface area for any input flow conditions. In the off-design mode, this area may not be equal to the actual value of the heat-transfer surface; if it is not, the heat load must be varied until the calculated area is equal to the actual area. This demand constraint cannot be established within the heat-exchanger model, because the model was set up to process the hot and cold flows at different stages in the analysis.

In general, these model demand constraints can be established in more than one way; it is up to the SALT user to decide how they will be met. For convenience, most of the constraints are evaluated within the model and assigned to the model parameter, denoted CONS. Thus, the difference between the steam-drum input enthalpy and that required by the model is denoted CONS. In this way, the constraints can be more easily set up within the SALT input.

3.4 FLOWS USED IN STEADY-STATE MODELS

As was indicated in Sec. 3.1, within the SALT code a flow is actually just a PL/I structure variable. At present, two main flow types -- GAS and FUEL -- are available for use with steady-state models. The GAS structure has already been shown in Sec. 3.1; this generic type is used for many different flows. Three other flows are provided -- STM, AIR, and LIQ -- that are structurally exactly the same as GAS (i.e., they are of the same generic type). These additional flows are provided for the sake of clarity within the SALT inputs when referring to the flows. For example, the heat-exchanger model (which actually requires flows of the generic type of GAS) may process a STM flow on its hot side and an AIR flow on its cold side.

The other flow type, FUEL, is used to represent a fuel flow and has the following PL/I structure:

```
1 FUEL,
  2 NAME CHAR(16),
  2 PROP,
    3 (TEMP,MASS,HHV) FLOAT(16),
  2 WEIGHTS,
    3 (C,H,O,N,S,CL,H2O,ASH) FLOAT(16);
```

where NAME represents the name of the flow as it is used within the SALT inputs, PROP is a substructure containing the elements, TEMP represents the flow's temperature, MASS is the mass flow rate, and HHV is the higher heating value. The substructure WEIGHTS is used to hold the weight fractions of carbon (C), hydrogen (H), oxygen (O), nitrogen (N), sulfur (S), chlorine (CL), water (H2O), and ash (ASH).

4 STEADY-STATE MODELS

All of the models used by SALT are compiled into run-time libraries that are searched (during a linking step) to include within the analysis only those models actually used in the system. In general, it is possible to have more than one library of models; different libraries may or may not be compatible in terms of the model names included and the flow types processed. Thus, for example, two different libraries could be generated with two different IN models to initiate two different flow types. In this case, the two different model libraries (and their corresponding interface files; see Sec. 7.1) should not be concatenated or used together at run time. At present, the models used in analyses of purely steady-state systems are included in one library, while those used in analyses of dynamic systems are included in another library.

The models described in this chapter are the most basic ones available within the SALT model libraries.* These models were developed for use in fossil-energy, open-cycle magnetohydrodynamic (MHD), fuel-cell, liquid-metal MHD, ocean thermal-energy conversion (OTEC), and other power plants where the most prominent flow type is GAS (of course, as indicated in Sec. 3.4, LIQ, AIR, or STM may be used instead). Most of these models process one or more flows of this type.

4.1 GENERIC COMBUSTOR/GASIFIER MODEL

The CB model, representing a generic combustor, requires three flows; the first two are inputs and the third is an output. The first flow represents the fuel input and is of the generic type FUEL. The second flow represents any oxidizing flow, while the third represents the combustion-gas output; these latter two flows are of the generic type GAS. The CB model, as a generic combustor, can also model a gasifier, where the output gas-flow conditions are at chemical equilibrium. Options are also provided for ash removal and potassium injection (used for MHD systems).

The parameters of the CB model are as follows:

HEAT_LOSS_FRAC -- Specified fraction of the thermal input (based on the higher heating value of the fuel) lost from the combustor due to heat loss.

FUEL_M -- Calculated value of the fuel mass after any ash removal.

FUEL_HHV -- Corrected higher heating value of the fuel after any ash removal.

ASH_M -- Calculated amount of mass removed from the fuel as ash.

*Further information on these models and their use, including declaration structures, is available in the authors' companion volume to the present report, *The Systems Analysis Language Translator (SALT): Programmer's Guide*, Argonne National Laboratory Report ANL/FE-85-04 (March 1985).

ASH_M_FUEL -- Calculated amount of mass left in the fuel as ash.

ASH_DET -- Specified weight fraction of the ash within the fuel after any ash rejection.

FUEL_HEAT_FORM -- Heat of formation of the fuel at a pressure of 1 atm and a temperature of 298.16 K.

H2O_M_FUEL -- Calculated amount of water left in the fuel.

SLURRY_CONC -- Calculated weight fraction of solid fuel to total weight of fuel (useful when CB is modeling a gasifier).

CARBON_BURNOUT -- Specified fraction of the carbon in the fuel that is actually burned; the rest of the carbon is carried over with any ash carry-over.

K_MASS -- Calculated weight of potassium in the output gas flow.

K_FRAC -- Specified weight fraction of potassium in the output gas flow (used to model potassium seed injection in MHD systems).

OX_M -- Calculated mass of oxygen needed for stoichiometric combustion of the fuel.

STOICH -- Calculated fraction of the mass of total oxygen in oxidizer flow to the mass, OX_M.

PRES_DROP_FRAC -- Specified fraction of the input oxidizer pressure representing the pressure drop through the combustor.

BC(8),BO(8),BG(8) -- Calculated elemental mass fractions for the fuel, oxidizer, and output gas flows, respectively.

4.2 GENERIC SYSTEM-COMPONENT MODELS

4.2.1 Compressor Model

The compressor model (CP) requires one pass-through flow of the generic type of GAS. A simplified off-design option is also provided. In the design mode, the model obtains the exit flow conditions by calculating an isentropic compression to a specified exit pressure and then corrects for a specified isentropic efficiency. In off-design use, a nondimensional mass factor is also calculated.

In the off-design mode, the model requires an initializing call to CPIN to obtain a table of pressure ratios vs. the mass factor (normalized by the design-point mass factor). During flow processing, the model then uses this table to calculate (based on the

inlet mass factor) the pressure ratio and, hence, the exit pressure. The model then proceeds as in the design mode.

The parameters of the CP model are as follows:

DDNAME -- Character string representing the file name of the off-design pressure-ratio-vs.-mass-factor table. This variable is not needed in the design mode. The information in this file consists of (1) an integer specifying the number of pressure-ratio values, followed by (2) the list of pressure-ratio values and by (3) the list of normalized mass-factor values.

MODE -- Character string representing mode, either "DESIGN" or "OFF-DESIGN."

EXIT_PRES -- Specified exit pressure for the design mode.

EFFICIENCY -- Specified isentropic efficiency of the compression process.

MASS_FACT -- Calculated mass factor for use in off-design calculations. This factor is an output in the design mode, but it must be an input in the off-design mode.

M_FACT -- Calculated value of the normalized mass factor used in off-design calculations. This variable is assigned the value 1 at the design point.

PRES_RATIO -- Calculated pressure ratio across the compressor.

4.2.2 Deaerator Model

The deaerator model (DEAR) requires two steam flows, the first of which is a pass-through flow representing not only one of the input flows, but also the output flow from the model. The DEAR model is a demand-type model, requiring that the exit flow be saturated.

The only parameter of the DEAR model is **QUAL**, the output flow quality from the model. For proper modeling of a deaerator, this parameter should be made to equal zero by imposing some system constraint.

4.2.3 Flash Model

This model (FLSH) represents a flash tank in which the entering flow is isenthalpically expanded through a given pressure drop. The model requires two flows; the first represents the incoming fluid (on input) or the vapor phase of the flash (on output). The second flow (an output flow) represents the liquid phase of the flash.

The parameters of the FLSH model are as follows:

PRES_DROP -- Specified pressure drop through the device.

QUAL -- Calculated quality of the input flow.

4.2.4 Feedwater-Heater Model

The closed feedwater heater modeled by FH incorporates desuperheating, condensing, and drain-cooling zones. The model is set up to process the hot flows -- extracted from the turbine and from higher-pressure feedwater heaters -- in one entry and the cold feedwater flow in another. The hot-flow entry (FHH) requires one pass-through flow (representing the turbine extraction flow on input and the drain-cooler exit flow on output) and one input flow (representing any cascaded flow from a higher-pressure feedwater heater). This hot-flow entry must be called before the cold flow entry (FHC), which requires one pass-through flow. All of the flows used within the FH model are of the generic type STM.

The parameters of the FH model are as follows:

SUBCOOL -- Specified amount of subcooling of the drain-cooler exit flow.

HEAT -- Calculated total amount of heat transferred from the hot flows to the cold flow.

AREA -- Calculated total surface area of the desuperheating and condensing regions of the feedwater heater.

TTD -- Calculated terminal temperature difference, defined as the difference in temperature between the hot-flow exit temperature from the condensing region and the cold-flow exit temperature from the desuperheating region.

TSAT -- Calculated hot-flow saturation temperature at the pressure within the condensing region.

FW_VEL -- Specified velocity of the cold feedwater flow through the condensing region.

DCTD -- Calculated drain-cooler temperature difference, defined as the temperature difference between the hot-flow exit temperature from the heater and the cold-flow entrance temperature.

HDP -- Specified flow pressure-drop fraction. (The pressure drop is equal to this parameter times the input pressure.)

CDP(3) -- Specified array of cold-flow pressure-drop fractions through the desuperheating section, CDP(1); the condensing section, CDP(2); and the drain-cooler section, CDP(3).

A(3) -- Calculated array of heat-transfer-surface areas for the individual feedwater-heater regions: desuperheater, 1; condenser, 2; and drain cooler, 3.

Q(3) -- Calculated array of heat-transfer values for the three regions of the heater.

U(3) -- Calculated array of heat-transfer coefficients for the three regions of the heater.

LMTD(3) -- Calculated array of log mean temperature differences for the three sections of the heater.

HTEMP(4) -- Calculated end-point temperatures of the hot flow between the three regions of the heater, where HTEMP(1) is the inlet temperature and HTEMP(4) is the exit temperature from the heater. Because HTEMP(2) and HTEMP(3) represent the hot-flow condensing-region temperatures, these two temperatures are both equal to the saturation temperature.

CTEMP(4) -- Calculated cold-flow temperatures between the three regions of the heater, where CTEMP(1) is the exit temperature and CTEMP(4) is the cold-flow inlet temperature.

4.2.5 Heater Model

The heater model (HT) requires one pass-through flow of the generic type GAS. The parameters of the model are as follows:

HEAT -- Specified heat added to the flow.

T_SET -- Specified exit temperature of the flow if set to a number greater than zero. This exit temperature determines the value of HEAT; if T_SET is set to zero, then HEAT must be input.

PRES_DROP_FRAC -- Fraction of the input flow pressure used as a pressure drop through the heater.

4.2.6 Heat-Exchanger Model

The heat-exchanger model (HX) is set up to process the hot flow in one entry (HXH) and the cold flow in another entry (HXC). Both entries require one pass-through flow of the generic type GAS. Either entry may be called first.

The model also includes options for calculating heat-transfer-surface areas using specified heat-transfer coefficients. These coefficients can be adjusted as functions of mass flow rate or temperature to simulate off-design changes. Thus, the model can be run off-design, but the surface areas (as calculated in the code) must be constrained to their design values outside the model (see CONS, below).

The parameters of the HX model are as follows:

MODE -- Specified character string taking the values of "DESIGN" or "OFF-DESIGN."

TYPE -- Specified character string taking the values "PARALLEL" or "COUNTER" to indicate that the heat exchanger is of either a parallel-flow or counter-flow configuration.

HEAT -- Specified heat transfer from the hot to the cold fluid (may be overridden if T_SET is set).

HEAT_FLUX -- Calculated average heat flux in the exchanger.

T_SET(2) -- An array of exit-temperature values; the first element is for the hot flow, the second, for the cold flow. Only one of these elements should be assigned a value (their default value is zero), and that one must correspond to the entry that is called first. If either element is assigned a value, then the heat transferred is calculated from the T_SET value rather than from the value set in HEAT; T_SET should be used only if the flow being assigned an exit temperature is definitely not in the two-phase region.

PRES_DROP_FRAC(2) -- Specified array of the fraction of input flow pressure used as a pressure drop through the device: hot flow, 1 and cold flow, 2.

U -- Calculated overall heat-transfer coefficient from the hot to the cold fluids.

AREA -- Total heat-transfer area, calculated in the design mode and specified in the off-design mode (however, see CONS).

INTEMP(2) -- Storage for the inlet fluid temperatures.

AVGTEMP(2) -- Calculated average temperatures of the hot and cold fluids.

ST(2) -- Calculated surface temperatures between the fluids and the wall.

LMTD -- Calculated log mean temperature difference between the fluids.

UR(3) -- Specified array of heat-transfer coefficients for hot fluid to wall (1), wall to cold fluid (2), and through wall (3).

UC(2) -- Specified array of correction factors for the UR(1) and UR(2) values in the off-design mode. If a value of UC exceeds 100, then the value of UR is adjusted as $UR \cdot (UC/AVGTEMP)^3$ or else as $UR \cdot (DM/MASS)^{UC}$, where MASS is the fluid-mass flow rate and DM is defined below. These parameters are used only in the off-design mode.

DM(2) -- Specified input values of the design mass flow rates (used only in the off-design mode, to correct the heat-transfer coefficients).

CONS -- Calculated off-design parameter representing the difference between the calculated and specified surface areas. In the off-design mode, this variable must be constrained to equal zero if the model is to yield the correct results.

PINCH_POINT -- Specified parameter representing the minimum value of the MEAN TDIF for which no error message indicating occurrence of a pinch-point violation is printed.

CAL(2) -- "Flags" used by the code to indicate when both hot and cold entries have been called and, thus, when the surface areas are to be calculated.

4.2.7 Flow-Initiator Model

The flow-initiator model (IN) requires one pass-through flow of the generic type GAS. The IN model also has two additional GAS flow-processing entries, INCYCL and INCOMP. The INCYCL entry calculates the differences in temperature, pressure, etc. of the flow between this entry and that of the INC entry. This entry is useful in setting up recycle loops.

The INCOMP entry is used to feed back to the INC entry the values of the INCOMP entry's GAS flow compositions. By calling this entry with the parameter ITER (incremented by one for each call), INCOMP will take its input flow composition and assign it to the model's COMP parameter. In this way, a simple fixed-point iteration scheme can be set up to converge on gas compositions in recycle loops by sweeping ITER from one to some maximum iteration number and by calling INC at the beginning of the loop and INCOMP at the end of the loop.

The parameters of the IN model are as follows:

ID -- Specified character-string variable representing the type of property code used in calculating thermodynamic properties of the flow.

ATOM(8) -- Calculated array of atomic-weight fractions of elements of the flow, if the ID is set to GAS.

T -- Specified temperature of the flow. If T is set to zero, the flow is assumed to be a condensable fluid and the saturation temperature is used. In this case, the enthalpy of the flow is determined using Q.

P -- Specified pressure of the flow.

H -- Calculated enthalpy of the flow.

S -- Calculated entropy of the flow.

Q -- Specified quality of the flow, used when T is set to zero. Flow quality Q may be greater than one (to represent superheating) or less than 0 (to represent subcooling).

V -- Specified velocity of the flow.

M -- Specified mass flow rate of the flow.

COMP -- Specified structure variable defining the molar fractions of the separate species that may be used with the GAS property code. The molar fraction of each species is specified as "X" followed by the species' chemical formula (e.g., XH₂, XCO₂, XNH₃).

SOL -- Structure variable representing the weight fractions of entrained solids within the gas flow. At present, SOL has only a single scalar (WTF) within its structure. This WTF represents the fraction of the flow's mass flow rate that is solid.

DT, DP, DV, DH, DM -- Calculated differences in T, P, V, H, and M between the flow originating from the INC entry and the flow entering the INCYCL entry.

ACC -- Termination criterion employed when the INCOMP entry and a SALT-defined parameter sweep over ITER are used to close a recycle loop over compositions. If the maximum difference in species concentrations between the INC and INCOMP entries is less than ACC, then ITER is set to 1000.

ITER -- Iteration counter used in the INCOMP entry.

ITERS -- Saved previous value of ITER.

PRINT -- Print switch used in the INCOMP entry.

4.2.8 Flow-Mixer Model

The flow-mixer model (MX) requires two flows. The first is a pass-through flow, representing one of the two input flows and also the output flow. The second flow is the second input flow. Both of these flows are of the generic type GAS.

The model does not require any parameters.

4.2.9 Steam-Condenser Model

Any of the condensible fluids (in addition to steam) may be used with the steam-condenser model (SC). The model requires only one pass-through flow, representing both the input flow and the condensed output flow. This flow is of the generic type STM. The energy extracted by the condensing process is saved in the POWER substructure.

The only parameter of the SC model is EXIT PRES, the specified exit pressure of the model. If EXIT PRES is set to zero, then the exit pressure is assumed to be equal to the inlet pressure.

4.2.10 Steam-Drum Model

The steam-drum model (SD) requires two flows, both of the generic type STM. The first is a pass-through flow, representing the two-phase input flow and, as an output, the downcomer flow. The second flow is an output flow, representing the saturated-steam flow. The model is a demand-type model; the input flow must be of a specified steam quality.

The parameters of the SD model are as follows:

QUAL -- Specified steam quality of the input flow.

CONS -- Difference between the enthalpy of the incoming flow and that required by the specified steam quality. This parameter is calculated by the model to aid in obtaining the correct input steam quality. Thus, CONS should be constrained to equal zero outside the model.

4.2.11 Flow-Splitter Model

The flow-splitter model (SP) requires one pass-through flow, representing the input flow and (on output) one of the two output flows. A second flow to the model represents the second output flow. Both of these flows are of the generic type GAS. The SP model provides options for splitting the flow not only by mass, but also by composition. Thus, the SP model can be used to model processes that split off specific species of the input flow.

The parameters of the SP model are as follows:

SPLIT_RATIO -- Specified fraction of the input flow mass split off into the second flow. If **SPLIT_RATIO** is set, then **SR** (see below) should not be used.

SPLIT_MASS -- Specified portion of mass flow rate split off into the second flow, if set greater than zero. If this variable is set to zero, then the mass flow rate split off is defined by **SPLIT_RATIO**. **SPLIT_MASS** must be less than the mass flow rate of the input flow.

SR -- Specified structure variable representing the split ratios by weight fractions of the species flow rates split off into the second flow. The elements of this structure are the same as those of the **COMP** substructure within the generic **GAS** flow, but without the "X" prefix (e.g., **AR**, **CH4**, **O2**, **SO2**). If the **SR** structure is specified, then the mass flow rate of the second flow is determined by the sum of the flow rates of the individual species. **SR** should not be specified if **SPLIT_RATIO** is used.

4.2.12 Pump Model

The pump modeled by **PUMP** handles liquids and requires one pass-through flow of the generic type **LIQ**. Rather than modeling the exact energy changes through the pump by iterating over the property procedures, **PUMP** uses an approximation. It calculates the power required as the change in pressure, divided by the density, times the efficiency. The exit enthalpy of the flow is obtained by adding this required power to the inlet flow enthalpy.

The parameters of the **PUMP** model are as follows:

EXIT_PRES -- Specified exit pressure of the flow.

EFFICIENCY -- Specified efficiency of the pump compression.

4.2.13 Fuel-Flow-Initiator Model

The fuel-flow-initiator model (**INF**) requires one pass-through flow of the generic type **FUEL**. The parameters of the **INF** model are as follows:

T -- Specified temperature of the fuel.

M -- Specified mass flow rate of the fuel.

HHV -- Specified higher heating value of the fuel.

WEIGHTS -- Structured variable representing the fuel composition by weight fractions. The **WEIGHTS** structure includes the following variables, where each variable represents the weight fraction of the substance in parentheses: C (carbon), H (hydrogen), O (oxygen), N (nitrogen), S (sulfur), Cl (chlorine), H₂O (water), and ASH (ash).

4.2.14 Fuel-Dryer Model

The fuel-dryer model (DRY) has two flow-processing entry points, DRYC and DRYH. The DRYC entry, which processes the fuel input and requires a pass-through flow of the generic type FUEL, performs the calculations involved in drying the fuel to a specified water fraction and calculates the heat energy required to vaporize the removed water. This entry must be called before the DRYH entry, which processes the hot-gas drying flow and should be of the generic type GAS.

The parameters of the DRY model are as follows:

H₂O_DET -- Specified weight fraction of water in the dried fuel.

H₂O_M -- Calculated mass of water removed from the fuel.

HEAT_REQUIRED -- Calculated energy required to vaporize the water mass removed.

4.2.15 Nozzle Model

The gas-nozzle model (NZ) requires one pass-through flow of the generic type GAS. The parameters of the NZ model are as follows:

EFFICIENCY -- Specified efficiency of the nozzle expansion, defined as the isentropic pressure drop necessary to accelerate the flow to the specified exit velocity, divided by the actual pressure drop.

EXIT_VEL -- Specified exit velocity of the gas flow.

PRINT -- Specified print switch; if set to a number greater than zero, this switch will print out the iterations within the model used in obtaining the isentropic exit pressure.

4.2.16 Diffuser Model

The gas-diffuser model (DF) requires one pass-through flow of the generic type GAS. The parameters of the DF model are as follows:

EXIT_VEL -- Specified exit velocity of the gas flow.

PRES_RECOVERY_COEF -- Specified value of the pressure-recovery coefficient, defined as the actual pressure drop across the diffuser divided by the difference in the total static pressure at the diffuser inlet.

PRINT -- Specified print switch; if set to a number greater than zero, this switch will print out the iterations within the model used in calculating the total inlet pressure.

4.2.17 Stack Model

The stack model (SK) requires one pass-through flow of the generic type GAS. The parameters of the SK model are as follows:

A_TEMP -- Specified ambient temperature at the stack exit.

A_PRES -- Specified ambient pressure at the stack exit.

4.3 TURBINE MODELS

4.3.1 Steam-Turbine Model

The steam-turbine model (ST) provides options for modeling a typical extraction stage, as well as inlet and exhaust stages. The model requires one pass-through flow and one output flow, representing any extracted flow from the turbine stage. This extracted flow is at the same thermodynamic conditions as the pass-through flow. Thus, the extracted flow is extracted at the exit of the turbine stage. Turbine trains with multiple extraction points would be modeled using multiple ST models. Both of the flows to the ST model are of the generic type STM.

The model provides for an off-design mode by calculating a flow factor in the design mode that is then used in the off-design mode. This flow factor represents a nondimensional flow rate that the turbine stage can pass. In the off-design mode, the parameter **CONS** (based on this flow factor) should be constrained to equal zero.

Tables of exhaust-loss enthalpy corrections for use in the exhaust stage may be read in by calling **STIN**. These tables define the value of the exhaust-loss enthalpy vs. the turbine inlet mass, normalized by dividing by a design-point mass. This exhaust-loss enthalpy is then added to the exit flow enthalpy from the turbine. (In design-mode calculations, this design-point mass flow rate is assigned the value of the inlet mass.)

For the inlet turbine stage in the design-mode, the exit pressure is calculated to give a required steam flow velocity. This required velocity is obtained from a specified turbine-wheel speed and a specified wheel-to-flow-velocity ratio. The efficiency is also calculated as a function of this wheel-to-flow-velocity ratio for both the design and off-design modes.

The parameters of the ST model are as follows:

DDNAME -- Specified character string representing the file name that STIN will read to obtain the table of exhaust-loss enthalpies.

MODE -- Specified character string representing either "OFF-DESIGN" or "DESIGN." In modeling throttle stages, the characters "IN" may also be appended to the end of the MODE string.

EXIT_PRES -- Specified exit pressure of the turbine.

EFFICIENCY -- Specified efficiency of the turbine expansion.

MECH_EFF -- Mechanical efficiency of the turbine. Any thermodynamic energy extracted from the turbine is multiplied by this efficiency to obtain the usable power.

SR -- Specified split ratio of the extracted flow. The mass flow rate of the extracted flow is equal to SR times the input mass flow rate.

EXT_MASS -- Specified extracted mass flow rate. If EXT_MASS is specified, it overrides SR. (EXT_MASS should be less than the inlet mass flow rate.)

FLOW_FACT -- Flow factor, calculated in the design mode and specified in the off-design mode.

EXHAUST_LOSS -- Specified or calculated value of the exhaust-loss enthalpy. If STIN has been called, this value is obtained from the tables; otherwise, the value may be specified as an input.

DM -- Specified value of the design-point mass flow rate in the off-design mode. In the design mode, this parameter is set equal to the rate of inlet mass flow to the turbine.

WV -- Specified turbine wheel-to-flow-velocity ratio, used only in the turbine-inlet stage.

WHEEL_SPEED -- Specified turbine-wheel speed, used only in the turbine-inlet stage.

CONS -- Calculated parameter in the off-design mode, representing a measure of the mass flow rate that the turbine stage can pass for the various inlet conditions. This parameter should be constrained to equal zero during off-design calculations.

VOL_FLOW_RATE -- Calculated volume flow rate through the turbine.

PRINT -- Specified switch used to print out iterations in the turbine-inlet-stage option during the calculation of exit pressure.

4.3.2 Gas-Turbine Model

The gas-turbine model (GT) requires one pass-through flow of the generic type GAS. A simplified off-design mode is also provided.

In the design mode, the exit flow conditions are calculated by means of an isentropic expansion to the specified exit pressure. The exit enthalpy is adjusted using the specified isentropic efficiency. A nondimensional mass factor is then calculated for use in off-design calculations.

In the off-design mode, an initial call to GTIN must be made to obtain a table of pressure ratios vs. normalized mass factors. This table is used during flow processing to obtain the pressure ratio for the calculated normalized mass factor. The exit flow conditions are then calculated by an expansion through this pressure ratio with the specified isentropic efficiency.

The parameters of the GT model are as follows:

DDNAME -- Specified character string representing the name of the file that contains the off-design pressure-ratio table (needed only in the off-design mode). Information contained in this file is in the following order: (1) an integer representing the number of pressure-ratio values, (2) the list of pressure ratios, and (3) the list of normalized mass factors.

MODE -- Specified character string taking the values of "DESIGN" or "OFF-DESIGN."

EFFICIENCY -- Specified isentropic efficiency of the turbine expansion.

MECH_EFF -- Specified mechanical efficiency; any thermal energy extracted through the turbine expansion is multiplied by this efficiency to obtain the useful mechanical-power output.

EXIT_PRES -- Specified design-point exit pressure (an output from the model in the off-design mode).

MASS_FACT -- Nondimensional mass factor calculated in the design mode and specified in the off-design mode.

M_FACT -- Calculated nondimensional mass factor normalized by dividing by the design-point mass factor.

PRDES -- Design-point pressure ratio calculated in the design mode and specified in the off-design mode.

PRES_RATIO -- Calculated pressure ratio across the turbine.

INIT -- A "flag" used by the code to initiate a reevaluation of the design-point mass factor on first entry to the model during the off-design mode. The design-point pressure ratio may not be at the maximum pressure ratio specified in the off-design pressure-ratio table; therefore, the design-point mass factor is adjusted to reflect what it would be at the maximum ratio in the table.

4.4 FUEL-CELL MODELS

4.4.1 Molten-Carbonate Fuel-Cell Model

The molten-carbonate fuel-cell model (MCFC) requires two pass-through flows of the generic type GAS. The first of these is the anode flow; the second, the cathode flow.

The parameters of the MCFC model are as follows:

CELL_CURRENT -- Specified current through each cell.

CELL_VOLTAGE -- Calculated cell voltage.

CELL_TEMP -- Specified average temperature of a cell.

STACK_VOLTAGE -- Calculated total voltage across the cell stack.

NO_OF_CELLS -- Specified total number of cells in the stack.

DELTA_VOLT -- Specified difference between the Nernst potential at the fuel cell exit and the cell voltage.

FUEL_UTIL -- Calculated value of the fuel utilization.

O2_UTIL -- Calculated value of the O₂ utilization.

CO2_UTIL -- Calculated value of the CO₂ utilization.

HF -- Calculated value of the overall isothermal heat of reaction.

E -- Calculated Nernst potential at the fuel-cell exit.

4.4.2 Solid-Oxide Fuel-Cell Model

The solid-oxide fuel-cell model (SOFC) requires two flows, both of the generic type GAS. The first flow represents the anode flow; the second, the cathode flow.

The parameters of the SOFC model are as follows:

CELL_CURRENT -- Specified current through each cell.

CELL_VOLTAGE -- Calculated cell voltage.

CELL_TEMP -- Specified average temperature of a cell.

STACK_VOLTAGE -- Calculated total voltage across the cell stack.

NO_OF_CELLS -- Specified total number of cells in the stack.

DELTA_VOLT -- Specified difference between the Nernst potential at the fuel-cell exit and the cell voltage.

FUEL_UTIL -- Calculated value of the fuel utilization.

O2_UTIL -- Calculated value of the O₂ utilization.

HF -- Calculated value of the overall isothermal heat of reaction.

E -- Calculated Nernst potential at the fuel-cell exit.

4.4.3 Phosphoric Acid Fuel-Cell Model

The phosphoric acid fuel-cell model (PAFC) requires two flows, both of the generic type GAS. The first flow represents the anode flow; the second, the cathode flow.

The parameters of the PAFC model are as follows:

CELL_CURRENT -- Specified current through each cell.

CELL_VOLTAGE -- Calculated cell voltage.

CELL_TEMP -- Specified average temperature of cell.

STACK_VOLTAGE -- Calculated total voltage across cell stack.

NO_OF_CELLS -- Specified total number of cells in stack.

DELTA_VOLT -- Specified difference between Nernst potential at fuel-cell exit and cell voltage.

FUEL_UTIL -- Calculated value of fuel utilization.

O2_UTIL -- Calculated value of O₂ utilization.

HF -- Calculated value of overall isothermal heat of reaction.

E -- Calculated Nernst potential at fuel-cell exit.

4.5 MAGNETOHYDRODYNAMIC-GENERATOR MODEL

The magnetohydrnamic-generator model (MG) simulates an MHD channel. The model has two entry points -- MGH, used to model the hot gas flow through the channel, and MGC, used to model the coolant flow through the channel. The MGH entry must be called before the MGC entry. Both entries require flows of the generic type GAS.

The parameters of the MG model are as follows:

AREA_INLET -- Calculated inlet flow area of the gas flow.

AREA_OUTLET -- Calculated outlet flow area of the gas flow.

B_FIELD -- Specified value of the magnetic field.

CONDUCTIVITY -- Calculated value of the electrical conductivity of the gas flow at the channel exit.

DELTA_LENGTH -- Specified length increment along the channel. Calculations along the channel are performed at discrete locations, DELTA_LENGTH apart.

EXIT_PRES -- Specified value of the cutoff pressure. Calculations along the channel terminate when the calculated pressure becomes less than this value. (Actual exit pressure will not necessarily attain this specified EXIT_PRES value.)

FARADAY_CURRENT -- Calculated value of the Faraday current.

FARADAY_FIELD -- Calculated value of the Faraday electric field.

FLOW_RATIO -- Calculated ratio of the channel length to the channel height at the exit.

FRACTION_HEAT_LOSS -- Calculated ratio of the heat loss to the coolant flow to the power produced by the channel.

FRACTION_PRES_LOSS -- Calculated ratio of the pressure drop along the channel to the inlet pressure for the gas flow.

FRICION_COEF -- Specified value of the friction coefficient (used in calculating the pressure drop along the channel).

HALL_FIELD -- Calculated value of the Hall field.

HALL_PARAMETER -- Calculated value of the maximum Hall parameter at the channel inlet or exit.

INVERTER_EFF -- Specified efficiency of the electrical inverter.

LENGTH -- Calculated length of the channel (a multiple of DELTA_LENGTH).

LOAD_FACTOR -- Specified value of the load factor along the channel.

MACH_NO_INLET -- Calculated value of the inlet-gas-flow Mach number.

MACH_NO_OUTLET -- Calculated value of the exit-gas-flow Mach number.

POWER_DENSITY -- Calculated value of the power density within the channel.

STANTON_NO -- Specified value of the Stanton number (used in calculating gas-side convective heat loss to the coolant flow).

WALL_TEMP -- Specified wall-temperature value (used in calculating heat loss to the coolant flow). This temperature should be between the coolant and gas-flow temperatures.

EXTRACTED -- Calculated value of the enthalpy extracted from the gas flow.

ABSORBED -- Calculated value of the enthalpy absorbed by the coolant flow.

4.6 COMPONENT MODELS FOR LIQUID-METAL SYSTEMS

4.6.1 Liquid-Metal Pipe Model

Liquid-metal flow through a pipe is modeled by MPIP, which requires one pass-through flow of the generic type LIQ. The parameters of the MPIP model are as follows:

FRIC_FAC -- Calculated friction factor of the flow within the pipe, established using simple Reynolds-number correlation.

VISC -- Specified viscosity of the flow within the pipe.

RE -- Calculated Reynolds number of the flow within the pipe.

FD -- Calculated pressure drop per unit length of pipe due to the frictional effects of the flow on the pipe.

FG -- Calculated pressure changes per unit length of pipe due to the effects of gravity on the mass of fluid within the pipe.

AREA -- Calculated flow area of the pipe, based on the inlet mass flow rate, density, and velocity.

DIAMETER -- Calculated pipe diameter, based on the calculated area.

LENGTH -- Specified length of the pipe.

GRAV_ANGLE -- Specified angle the pipe makes with the gravitational field.

4.6.2 Liquid-Metal Nozzle Model

The liquid-metal nozzle model (MNOZ) requires one pass-through flow of the generic type LIQ. The parameters of the MNOZ model are as follows:

EFFICIENCY -- Specified efficiency of the nozzle, defined as the change in velocity heads divided by the change in pressure across the nozzle.

EXIT_VELOCITY -- Specified exit velocity from the nozzle.

LENGTH -- Specified length of the nozzle (used in determining gravitational effects on the pressure changes across the nozzle due to the mass of fluid within the nozzle).

GRAV_ANGLE -- Specified angle that the nozzle makes with the gravitational field.

4.6.3 Liquid-Metal Diffuser Model

The liquid-metal diffuser model (MDIF) requires one pass-through flow of the generic type LIQ. The parameters of the MDIF model are as follows:

EXIT_VELOCITY -- Specified exit flow velocity.

EFFICIENCY -- Specified efficiency of the diffuser (defined as diffuser pressure rise divided by change in velocity heads).

LENGTH -- Length of the diffuser, used in calculating pressure changes due to gravitational effects on the liquid mass. (This pressure change is added to that due to the diffuser efficiency.)

GRAV_ANGLE -- Angle that the diffuser makes with respect to the gravitational field (in degrees). At GRAV_ANGLE=90, no gravitational effects are present.

4.6.4 Liquid-Metal Magnetohydrodynamic-Generator Model

The two-component liquid-metal MHD-generator model (MMHD) requires two pass-through flows of the generic types GAS and LIQ. The first flow represents the predominant gaseous component of the two-component flow, while the second represents the liquid component.

The parameters of the MMHD model are as follows:

EFFICIENCY -- Specified isentropic expansion efficiency of the generator.

EXIT_PRES -- Specified exit pressure from the generator.

SLIP_RATIO -- Specified ratio of the gas velocity to the liquid velocity at the exit of the generator.

TEMP_DIFF -- Specified difference between the liquid temperature and that of the gas at the exit of the generator.

LENGTH -- Length of the generator (used in calculating the gain or loss in energy due to the effects of gravity on the mass of fluid within the generator).

GRAV_ANGLE -- Specified angle the generator makes with the gravitational field.

VOID_FRACTION -- Calculated void fraction at the exit of the generator.

4.7 TWO-COMPONENT LIQUID/GAS SEPARATOR MODEL

The two-phase, two-component liquid/gas separator model (SEPR) requires two pass-through flows and two output flows, all of the generic type GAS. The first pass-through flow represents the predominant gaseous component, while the second represents the liquid component. The first of the output flows represents any gaseous carry-over flow that leaves with the liquid pass-through flow. The second output flow represents any liquid carry-over flow that leaves the separator with the gaseous pass-through flow.

The parameters of the SEPR model are as follows:

VELOCITY_HEAD_RATIO -- Specified ratio of the square of the liquid velocity out of the separator to the square of the liquid velocity into the separator (used only when the separator is run in the specified efficiency mode).

PRES_DROP(2) -- Specified array of pressure drops for the gaseous and liquid flows through the separator.

VOL_RATIO -- Calculated ratio of the volume flow rate for the gas to that for the liquid.

LIQ_CO -- Specified fraction of the liquid flow rate carried over with the gas flow.

GAS_CO -- Specified fraction of the gas flow rate carried over with the liquid flow.

EFFICIENCY -- Calculated ratio of the fraction of liquid mass to total mass times the velocity head ratio.

VAPOR_CO -- Calculated mass of liquid carried over with the gas flow due to the vaporization of the liquid. This parameter, calculated when VAPOR_INC is set to "YES," is used only to indicate how much of the liquid vapor is carried over. The actual mass is not combined with the exiting gas flow or subtracted from the liquid flow.

VAPOR_INC -- Switch used to indicate whether or not liquid vapor carry-over is to be calculated.

HEAT_REJECTED -- Calculated heat that would be lost from the liquid if vapor carry-over occurred.

4.8 TWO-PHASE COMPONENT MODELS

4.8.1 Two-Phase Mixer Model

The two-phase, two-component mixer model (TPMX) requires two pass-through flows, the first being the gaseous component and the second, the liquid component. The parameters of the TPMX model are as follows:

PRES_OUT_OPTION -- Specified character string defining an option for calculation of the exit pressure as a weighted average of the gas and liquid inlet pressures. If mix is equal to "MIX," this option is used; otherwise, the output flow pressure is taken as the minimum inlet flow pressure minus any specified pressure drop.

PRES_DROP -- Specified pressure drop through the mixer.

DP_FRAC -- Specified fraction of the minimum or weighted average inlet pressure, used as an additional pressure drop through the mixer.

SLIP_RATIO -- Specified ratio between the gas and liquid flow velocities.

TEMP_DIFF -- Specified difference between the gas temperature and that of the liquid.

PRES_DIFF_IN -- Calculated difference between the gas inlet pressure and that of the liquid.

VOID_FRACTION -- Calculated void fraction of the exit flow.

4.8.2 Two-Phase Nozzle Model

The two-phase, two-component nozzle model (TPNZ) requires two pass-through flows of the generic types GAS and LIQ. The first flow represents the gaseous component and the second, the liquid component. The parameters of the TPNZ model are as follows:

EFFICIENCY -- Specified efficiency of the nozzle, defined as the ratio of the actual change in enthalpy across the nozzle to the isentropic enthalpy change.

EXIT_PRES -- Specified exit pressure from the nozzle.

SLIP_RATIO -- Specified ratio of the gas velocity to the liquid velocity at the nozzle exit.

TEMP_DIFF -- Specified difference in temperature between the liquid and the gas at the nozzle exit.

LENGTH -- Specified length of the nozzle.

GRAV_ANGLE -- Specified angle between the nozzle and the gravitational field.

VOID_FRACTION -- Calculated void fraction of the flow at the nozzle exit.

4.8.3 Two-Phase Diffuser Model

The two-phase, two-component diffuser model (TPDF) requires two pass-through flows, the first representing the gaseous-phase component and the second the liquid-phase component. Both of these flows are of the generic type GAS or LIQ.

The parameters of the TPDF model are as follows:

MODE -- Specified character string taking on the values " " or "SPEC-EFF." If "SPEC-EFF" is not set, then the efficiency of the diffusion process is calculated within the code (based on the void fraction of the flow).

EXIT_VELOCITY -- Specified exit velocity of the liquid flow.

SLIP_RATIO -- Specified ratio of the gas velocity to the liquid velocity.

EFFICIENCY -- Specified efficiency of the diffuser, defined as the ratio of change in pressure across the diffuser to the change in velocity head across the diffuser.

LENGTH -- Specified length of the diffuser (used in calculating additional pressure changes due to gravity).

GRAV_ANGLE -- Specified angle the diffuser makes with the gravitational field.

VOID_FRAC_IN -- Calculated inlet void fraction.

VOID_FRAC_OUT -- Calculated exit void fraction.

4.9 SYSTEM MODEL

The system model (SYST) calculates the total power put in, produced, consumed, and lost by the system. The model does not require any flows.

The parameters of the SYST model are as follows:

POWER_HEAD_PTR -- Specified pointer to the linked list of model power substructures.

FLOW_HEAD_PTR -- Specified pointer to the linked list of model flow substructures.

NET -- Calculated net power produced by the system (total power produced minus total power consumed).

EFFICIENCY -- Calculated system efficiency based on total power input, net power, and auxiliary power. If total input power is zero, **EFFICIENCY** is also set to zero.

AUXILIARY -- Specified value of any auxiliary-power requirements. Auxiliary power is subtracted from net power in calculating efficiency.

UNITS -- Character string that indicates SI for output in SI units; otherwise, British units are used for output.

5 EXAMPLES USING STEADY-STATE MODELS

Unlike the examples provided in Chapter 1, all examples discussed in this chapter conform to the actual models available in the SALT library. The reader may find it useful, in going through these examples, to refer to the relevant sections in Chapter 4 to review the documentation on particular models.

5.1 SIMPLE SYSTEM CONFIGURATIONS

5.1.1 Simple Steam-Flow System

The first example demonstrates all the inputs necessary to run a very simple system configuration. Consider a steam/water flow, heated by a heater (HT) and then run through a steam turbine (ST). The system consists of one flow type, STM, and two model types, HT and ST; the following PROCESS statement represents this system:

```
PROCESS STM_1-> HT_1 ST_1 ->STM_EXT
```

A label has been appended to the models and flows to satisfy the requirement that all models and flows must have labels. The ST model requires an additional output flow, which has been labeled "EXT." As a general rule, all flows in a system configuration should originate within a component model. Thus, since the HT model does not generate a flow (its only flow is a pass-through flow), an inlet model (IN) is used to generate a flow for it. The complete system configuration for this example is then represented by the following:

```
PROCESS STM_1-> IN_1 HT_1 ST_1 ->STM_EXT
```

Before this statement can be executed, the properties procedures must be initialized. This is accomplished by the statement

```
PROCESS GP_1:IN
```

After the component models have been executed, the results of the analysis can be printed out by calling all of the output entries of all the models:

```
PROCESS NULL-> IN_1:OUT HT_1:OUT ST_1:OUT
```

The output entries require no flow arguments. Thus, the last-used pass-through flow ("STM_1") must be nullified; otherwise, it would be passed to the output entries. The NULL flow is used to turn off the pass-through flow. This PROCESS statement can be written more conveniently as

```
PROCESS NULL-> *_*:OUT
```

The use of `*_*:OUT` implies that each model used in the system analysis that has an `":OUT"` entry is to be called. The complete system-configuration specification for this problem (forming the contents of the SALT input file, `STRUCT`) is as follows:

```
PROCESS GP 1:IN
PROCESS STM 1-> IN_1 HT_1 ST_1 ->STM_EXT
PROCESS NULL-> *_*:OUT
```

The only other data needed for this problem are the values of the model input parameters, which can be specified by the use of the `DATA` statement. A typical example of this statement for this problem would be the following:

```
DATA
IN 1.PARM .ID='H2O'; .T=0; .P=150.; .M=20.; .Q=0.0;
HT_1.PARM .HEAT=1E6;
ST_1.PARM .EXIT_PRES=10.; .EFFICIENCY=0.88;
```

Here, the parameters for the `IN` model define the initial steam-flow conditions upstream of the `HT` model. The specification of the temperature as zero ($T = 0$) implies that the flow is saturated or in the two-phase region; the values of the pressure, P , and quality, Q , determine the temperature, as well as the other flow properties of enthalpy and entropy. The statement `HT.PARM.HEAT=1E6` specifies that one megawatt of energy will be transferred to the steam flow before entering the `ST` model; there, the steam flow will be expanded to 10 atm at an isentropic efficiency of 88%, as specified by the `ST` model parameters. The other parameters for these models will be taken as their default values (defined within the `INTF` file). In particular, if the `ST` parameter (`SR`) is defaulted to zero, then no extraction flow will exist; however, this flow must still be shown in the `PROCESS` statement.

5.1.2 Inclusion of a System Model

Suppose that the steam flow from the `ST` model of the previous section is condensed in a steam-condenser model (`SC`) and then pumped back to the inlet pressure in a pump (`PUMP`). The processing of the steam flow from the inlet, `IN`, through the `HT`, `ST`, `SC`, and `PUMP` models is represented as follows:

```
PROCESS STM_1-> IN_1 HT_1 ST_1 ->STM_EXT SC_1 PUMP_1
```

The `SC` and `PUMP` models were simply added to the end of the `PROCESS` statement. The flow passing through these models is that of `STM_1` and not `STM_EXT`, because the last pass-through flow is that of `STM_1`.

Suppose that in addition to the component powers generated by the `ST`, consumed by the `PUMP`, and rejected (as heat) from the system by the `SC`, the net power is also desired. The net power could be calculated by using the `PLI` key word and coding the appropriate `PL/I` statements (with reference to the model `POWER` substructures). However, the net power can be more easily determined using the `SYST` model. This model does not require any flows, so it should be specified after the `STM_1` pass-through flow has been nullified.

The entire input specified in the STRUCT file for this problem would be as follows:

```
PROCESS GP_1:IN
PROCESS STM_1-> IN_1 HT_1 ST_1 ->STM_EXT SC_1 PUMP_1
PROCESS NULL-> SYST_1 *_*:OUT
```

In this case, the additional model input parameters for the SC, PUMP, and SYST models must be specified along with the other model parameters in the DATA statement. No input parameters are required for the SC model. For the PUMP, the EXIT_PRES and EFFICIENCY parameters must be specified. Because the SYST model processes the substructures of all the models, the head pointers of the linked list of such substructures need to be assigned to the SYST parameters of POWER_HEAD_PTR and FLOW_HEAD_PTR. The names of these head pointers are defined within the INTF file; at present, they are the same as the SYST parameter names themselves. Thus, the following would be added to the DATA statement of the system considered in Sec. 5.1.1:

```
PUMP_1.PARM .EXIT PRES=250.; .EFFICIENCY=0.72;
SYST_1.PARM .POWER_HEAD_PTR=POWER_HEAD_PTR;
           .FLOW_HEAD_PTR=FLOW_HEAD_PTR;
```

5.1.3 Inclusion of a Demand Model Constraint

Figure 12 depicts a system consisting of a water/steam flow (LIQ) that is run through a heater (HT) and then through a steam drum (SD). The generated steam flow from the drum is processed by a steam turbine (ST), a condenser (SC), a water pump (PUMP), and a mixer (MX), where it is mixed with the downcomer flow (LIQ) from the SD model.

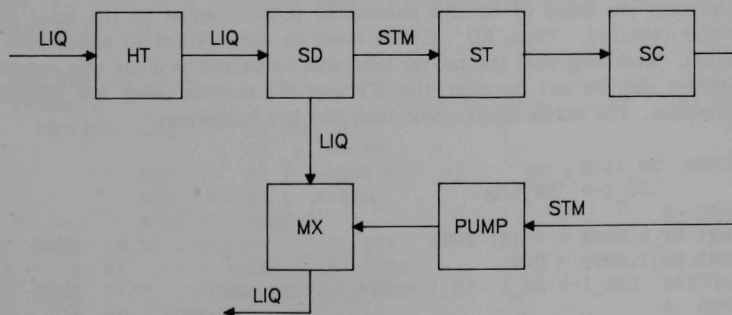


FIGURE 12 Simple Steam-Plant System

This system configuration is easily represented by the following PROCESS statement, using the general rules for representing pass-through, input, and output flows:

```
PROCESS LIQ_1-> IN_LIQ HT_1
              SD_1 ->STM_1
STM_1-> ST_1 ->STM_EXT SC_1 PUMP_1
LIQ_1-> MX_1 <-STM_1
```

Here, the model IN_LIQ has been added to initiate the LIQ_1 flow. Also, although it is not shown in Fig. 12, the extraction flow from the ST model has been included in the PROCESS statement.

To complete the inputs for this example, the properties procedures should be initialized, the SYST model may be added, and the outputs may be printed by use of the * *:OUT. The complete system configuration specified in the STRUCT file would be as follows:

```
PROCESS GP_1:IN
PROCESS LIQ_1-> IN_LIQ HT_1
              SD_1 ->STM_1
STM_1-> ST_1 ->STM_EXT SC_1 PUMP_1
LIQ_1-> MX_1 <-STM_1
NULL-> SYST_1 * *:OUT
```

As was indicated in Chapter 3, some models are of a demand type and require specific constraints to be added to the SALT input. The SD model is of such a type; it requires a specified input flow quality in order to work properly. This specified quality is a constraint that may be met in many different ways. In the system discussed in Secs. 5.1.1 and 5.1.2, for example, the heat load in the heater might be varied until the correct steam quality is reached, or (for a fixed heater load) the input steam-flow rate might be varied. In either case, a suitable VARY statement would be needed. In order to obtain the correct inlet steam-drum quality corresponding to the required steam-drum quality, the SD model sets the value of the SD parameter (CONS) equal to the inlet enthalpy minus the value required. Thus, SD_1.CONS must be constrained to equal zero within some subsystem. Assuming that the heater load will be varied to meet this constraint, a simple subsystem can be set up over the HT and SD models using the SYSBEG and SYSEND delimiters. The entire input would look like the following:

```
PROCESS GP_1:IN
              LIQ_1-> IN_LIQ
SYSBEG A
  VARY HT_1.HEAT = * 1E1 20E6
  CONS SD_1.CONS = 0.0
  PROCESS LIQ_1-> HT_1 SD_1 ->STM_1
SYSEND A
PROCESS STM_1-> ST_1 ->STM_EXT
              SC_1 PUMP_1
              LIQ_1-> MX_1 <-STM_1
              NULL-> SYST_1 * *:OUT
DATA .
      .
```

Here, the VARY statement instructs SALT to vary the HT parameter HEAT between 10 W and 1 MW until the constraint on the steam-drum parameter, CONS, is established. The "*" used in the VARY statement tells the SALT systems code to start its iterations with the current value of HT 1.HEAT; this value would be either its default value or the value specified in the DATA statement.

The information in the DATA statement should reflect the model parameters for the IN_LIQ, HT, SD, ST, SC, WP, and MX models. The MX does not require any parameters, and the SD requires only the specified inlet steam quality (QUAL), for which 0.2 would be a typical value. The other models can use the same DATA statement as was presented in Sec. 5.1.2.

5.1.4 Inclusion of User-Imposed Constraints

In Sec. 5.1.3, a constraint was added to accommodate the requirements of a demand model. Such constraints must be added to the system in order that such models work properly. However, additional constraints may be added to represent various user-imposed system requirements. These constraints may be added, as additional VARY and CONSTRAIN statements, to the subsystems used to establish the demand model constraints (provided, of course, that the parameters varied and the constraints both lie within those subsystems). Alternatively, user-imposed constraints may be established by setting up additional subsystems using additional SYSBEG and SYSEND statements. In general, any number of such subsystems can be set up to establish other system constraints. A given subsystem may even contain other subsystems.

Suppose that, besides the constraint on steam-drum quality, a constraint on the total power output is required:

```
SYST_1.SPOWER.PRODUCED=10E6
```

To satisfy this constraint, the inlet mass flow rate (LIQ) will be varied. This is easily represented as follows:

```
PROCESS GP_1:IN
SYSBEG A
  PROCESS LIQ_1-> IN_LIQ HT_1
                  SD_1  ->STM_1
                  STM_1-> ST_1  ->STM_EXT SC_1  WP_1
                  LIQ_1-> MX_1  <-STM_1
                  NULL-> SYST_1
  VARY IN LIQ.PARM.M= * 10 100
      ; HT_1.PARM.HEAT= * 1E1 20E6
  CONS SYST_1.SPOWER.PRODUCED=10E6
      ; SD_1.CON=0.0
SYSEND A
PROCESS NULL-> * *:OUT
DATA .
.
```

Here, the constraint on the system power was simply added to the subsystem used to establish the SD demand constraint by enlarging that subsystem from the IN_LIQ model down to the SYST_1 model. The IN_LIQ model was included within the subsystem, because one of its parameters is varied within the subsystem. The SYST_1 model was included within this subsystem so that the system power produced would be calculated at each iteration. The call to `*_*:OUT` is placed outside of the subsystem loop; otherwise, output would be produced for every iteration.

The same system problem can also be handled by the use of two nested subsystems, as follows:

```

PROCESS  GP_1:IN
SYSBEG  B
  VARY  IN_LIQ.PARM.M = * 10 100
  CONS  SYST_1.SPOWER.PRODUCED = 10E6
  PROCESS LIQ_1-> IN_LIQ
  SYSBEG  A
    VARY  HT_1.PARM.HEAT = * 1E1 20E6
    CONS  SD_1.CONS = 0.0
    PROCESS LIQ_1-> HT_1 SD_1 ->STM_1
  SYSEND  A
  PROCESS STM_1-> ST_1 ->STM_EXT SC_1 PUMP_1
    LIQ_1-> MX_1 <-STM_1
    NULL-> SYST_1
  SYSEND  B
PROCESS  NULL-> *_*:OUT
DATA    .
        .
        .

```

In this case, the iterations within the inner subsystem (labelled A) to establish the steam-drum constraint will be performed for each iteration of the outer subsystem to establish the system power constraint.

In a problem such as this, when the MINIMIZE statement (see Sec. 5.1.5, below) has not been specified, the number of constraints must equal the number of variables included in the VARY statements. Otherwise, the problem will be either overdetermined or underdetermined. Such equality-constrained problems make use of an n-dimensional hybrid equation solver. This equation solver is reasonably robust, but (depending on the problem) it may fail occasionally. If a failure occurs, it might be possible to decompose the problem into nested sets of smaller-dimensional problems or to try a new initial guess to find a solution. Sometimes, some of the algorithm's controlling parameters need to be adjusted. These controlling parameters may be assigned values by the use of the SWITCH statement. Like the VARY and CONSTRAIN statements, the SWITCH statement should lie between the SYSBEG and SYSEND statements to which it refers. A typical example might be the following:

```

SWITCH  MAXIT=50  DEL=1E-4  ACC=1E-2

```

where MAXIT is the maximum number of iterations that will be used to perform the task within the subsystem, and DEL represents a measure of how the variables within the subsystem will be perturbed in determining the gradients of the constraints. A DEL of $1\text{E-}4$ means each variable will be perturbed by a factor of 10^{-4} times the value of the variable. The value of DEL should be made as small as possible to represent the gradients accurately, but not so small that round-off error hinders the calculations. ACC is the termination criterion. Whenever the sum of the squares of the constraint violations is less than ACC, the iterations are terminated. The equation solver will attempt to scale the variables and constraints internally, so all constraints will usually converge to zero uniformly.

5.1.5 Inclusion of an Optimization Problem

Suppose that, rather than constraining the plant power produced, the maximum plant power is to be obtained as a function of the inlet steam pressure. The MINIMIZE statement may be used to set up this problem as follows:

```
PROCESS GP_1:IN
SYSBEG A
  PROCESS LIQ_1-> IN_LIQ HT_1
                    SD_1 ->STM_1
                    STM_1-> ST_1 ->STM_EXT SC_1 PUMP_1
                    LIQ_1-> MX_1 <-STM_1
                    NULL-> SYST_1
  VARY IN_LIQ.PARM.P = * 100 200
      ; HT_1.PARM.HEAT= * 1E1 20E6
  CONS SD_1.CON=0.0
  MINIMIZE -SYST_1.SPOWER.PRODUCED
SYSEND A
PROCESS NULL-> * *:OUT
DATA .
.
.
```

When the MINIMIZE statement is used, the number of variables in the VARY statement should exceed the number of equality constraints; inequality constraints also may be included. For instance, if an upper limit of 600 K on the heater exit temperature is required, the constraint would be stated as follows:

```
CONS HT_1.FLC.TEMP<600
```

It is permitted to have more inequality constraints than variables when the MINIMIZE statement is used.

Optimization problems are inherently more difficult to solve than pure-equality-constrained problems. The algorithm that is used in solving these nonlinearly constrained optimization problems is that of M.J.D. Powell.¹⁰ For the algorithm to work at its best, the constraints and the objective function, as defined by the MINIMIZE statement, should be continuously differentiable functions. As with the equation solver, failure may occur

when attempting some problems. Again, variation of the initial guesses, decomposition of the problem, or variation of the optimizer parameters could resolve the problem. The same parameters -- ACC, MAXIT, DEL, and PRINT -- are available for the optimizer as for the equation solver and have roughly the same meanings. Now, however, the iterations terminate when the gradient of the objective function plus the sum of the absolute values of the Lagrangian multipliers times the constraint violations is less than ACC. In decomposing such a problem, inner iterative loops that are pure-equality-constrained problems may be set up. (There is no restriction against using both equation-solver and optimizer loops in the same problem.)

Another difficulty associated with optimization problems is that only local minimums may be found. No good techniques currently exist for finding the global minimum of a general nonlinear problem. Where such a situation is suspected, parameter sweeps might be necessary to confirm the results of the optimization.

5.1.6 Inclusion of a Parameter Sweep

As was indicated in Sec. 5.1.5, parameter sweeps can be a better alternative than optimizations, at least when the problem dimension is small. Consider again the system discussed in Sec. 5.1.3. Suppose it is desired to determine the output power (in fact, all output parameters) as a function of the inlet liquid pressure as it is swept over 100 to 200 atm (in increments of 10 atm). This task is symbolized using the following SWEEP statement:

```
SWEEP IN_LIQ.P = 100 TO 200 BY 10
```

This statement would be included within its own separate subsystem as follows:

```
PROCESS GP_1:IN
SYSBEG B
  SWEEP IN_LIQ.P = 100 TO 200 BY 10
  PROCESS LIQ_1-> IN_LIQ
  SYSBEG A
    VARY HT_1.PARM.HEAT = * 1E1 20E6
    CONS SD_1.CONST = 0.0
    PROCESS LIQ_1-> HT_1 SD_1 ->STM_1
  SYSEND A
  PROCESS STM_1-> ST_1 ->STM_EXT SC_1 PUMP_1
    LIQ_1-> MX_1 <-STM_1
    NULL-> SYST_1 *_*:OUT
  SYSEND B
DATA .
.
.
```

In this case, the PROCESS NULL-> *_*:OUT is placed within the SYSBEG-SYSEND subsystem delimiters, so the output is produced for each value of the inlet liquid pressure.

5.1.7 Inclusion of a Feedback Loop

In Sec. 5.1.3, the liquid flow entered and then left the system but did not form a closed loop. Many systems do include closed loops. We now consider a system in which an additional pump has been added after the mixer and in which the liquid flow closes upon itself (see Fig. 13).

No real starting point for the liquid flow exists in such a system, so one is created artificially by tearing the flow path at some point and using the initiator model. The inlet conditions (flow conditions out of the initiator) are then varied until they match those out of the last model at the point of the tear. For example, if one tears the flow between the pump and the heater model and includes an initiator model at that point, the PROCESS statement for the configuration may be written as follows:

```
PROCESS LIQ_1-> IN_LIQ HT_1 SD_1 ->STM_1
        STM_1-> ST_1 ->STM_EXT_SC_1 PUMP_1
        LIQ_1-> MX_1 <-STM_1 PUMP_2
```

One now needs to examine the flow conditions out of the PUMP_2 model and ask what conditions at the IN_LIQ model must be varied in order to make these two flows equal. Since all of the flow mass leaving the IN_LIQ model eventually leaves the PUMP_2 model, the mass flow rates are the same for any inlet flow rate. The exit pressure from the PUMP_2 is an assigned input value to that model, so assigning that same value to the IN_LIQ.P parameter will make the pressures match for the two flows. The only other condition to be matched is temperature. The temperature out of the pump is not known; thus, to establish equality of temperature between the outlet flow of the pump and the IN_LIQ flow, one could vary the IN_LIQ flow temperature as follows:

```
VARY IN_LIQ.T = 400 300 500
```

until the following condition is met:

```
CONS PUMP_2.FLC.TEMP = IN_LIQ.T
```

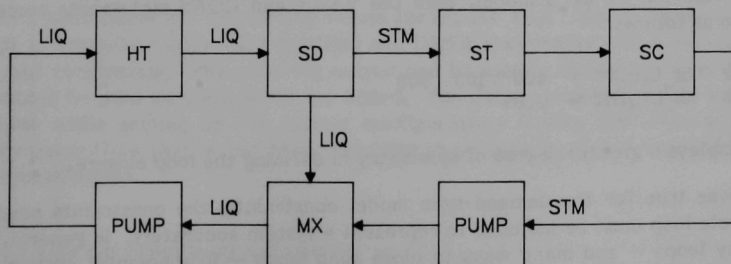


FIGURE 13 Simple Steam-Plant System with Feedback Loop

Then the system inputs would look like this:

```

PROCESS  GP_1:IN
SYSBEG  A
  VARY  IN_LIQ.T = * 300 500
  CONS  PUMP_2.FLC.TEMP = IN_LIQ.T
PROCESS  LIQ_1-> IN_LIQ HT_1 SD_1 ->STM_1
        STM_1-> ST_1 ->STM_EXT SC_1 PUMP_1
        LIQ_1-> MX_1 <-STM_1 PUMP_2
SYSSEND  A
PROCESS  NULL-> SYST_1 *_*:OUT
DATA    .
        .
        .

```

The DATA statement is similar to that used in Sec. 5.1.3, but now it is important to set the IN_LIQ.P parameter equal to the value assigned to PUMP_2.EXIT_PRES.

The point at which a flow stream is torn in order to close a feedback loop can affect which variables need to be varied to close the loop. By choosing the tearing point after the pump in the above example, one could match the pressures without any VARY-CONS statements. Similarly, by choosing this point as the tearing point, the flow is (for sufficiently high pump pressure) subcooled, and it is possible to vary the temperature. For other recycle loops, if it is not known beforehand whether or not a two-phase region might be entered, the enthalpy rather than the temperature should be varied. (The inlet parameter PARM.T would be set to zero and the parameter PARM.Q, representing the flow quality, would be varied. Subcooling and superheating are represented by varying Q below 0.0 and above 1.0.)

In order to more clearly represent the recycle loops using the PROCESS statement, an additional entry to the IN model may be called to represent the "back door" to the model. This entry, denoted "CYCL," also calculates the difference between the flow entering this entry and that originating from the IN model. This difference is stored in the variables DT, DP, DH, DM, and DV, representing the difference in temperature, pressure, enthalpy, mass, and velocity, respectively. In the above problem, for example, if the model entry IN_LIQ:CYCL had been specified in the PROCESS statement after the PUMP_2 model, then the VARY and CONS statements could have been written as follows:

```

VARY  IN_LIQ.T = 400 300 500
CONS  IN_LIQ.DT = 0.0

```

This form displays a greater degree of symmetry in defining the loop closure.

As was true for the demand-type model constraints, the constraints needed to close a recycle loop must be included to represent a system accurately. In general, there may be many loops -- and many ways to close such loops -- in a complex system. The closure also depends on the variables within the flow itself. Thus, in recycle gas flows, it may also be necessary to close on species concentrations, in addition to pressure, temperature, and mass flow rate.

5.2 SUMMARY OF THE INPUT-FORMATION PROCESS

The general process by which the inputs for a system problem are formed usually entails the following stages:

1. For the given system configuration, appropriately label the models and flows (consistent with each model's requirements as to pass-through, input, and output flows).
2. Formulate the PROCESS statement for the system, tearing flows for recycle loops and inserting appropriate flow initiators as necessary.
3. Consider any demand-type models included in the system and decide how their demand constraints will be satisfied (i.e., choose the parameters to be varied).
4. Decide on each recycle loop and how it will be closed. (Loop closure sometimes depends on model demand constraints, and these constraints sometimes depend on closure.)
5. Add any additional constraints and their establishing parameters that may be needed to define system requirements (i.e., user-defined constraints).
6. Work out the subsystem breakdown for implementing all of the constraints. This, of course, may be accomplished while one is adding the constraints and their establishing parameters. (One choice of a subsystem breakdown is simply to include all constraints in one subsystem, which is the entire system.)
7. Add any calls to the property-initialization procedures, system models, output entries, and the DATA statement.

As the system configuration becomes more complicated, it can become difficult to obtain a reasonable set of starting values for all the VARY parameters. It may then be useful to formulate only the PROCESS and DATA statements and to run the analysis without any constraints. The resulting output can be used to determine what parametric values should be used as constraints are added. The output may also reveal a conceptual error made while setting up the system configuration; finding the error at this point saves computer time that would have been lost in performing iterations on an incorrect system specification.

5.3 ANALYSIS OF POWER-PLANT SYSTEMS

Several more detailed examples will now be given. These examples consist of a conventional fossil/steam power plant, an open-cycle MHD plant, a solid-oxide fuel-cell plant, and a liquid-metal MHD plant.

5.3.1 Fossil/Steam Power Plant

Figure 14 shows a simple fossil/steam power plant that employs air preheating, steam superheating, and reheating; the plant is equipped with six steam-turbine models (to account for all extraction points) and four feedwater heaters. With the models and flows labeled as shown in the figure, we can write the PROCESS statement. (By starting the steam flow before the steam drum, there is only one recycle loop.) Starting with the steam flow, the PROCESS statement would be as follows:

```

PROCESS
LIQ_1->    IN_H2O  SD_1 ->STM_1
STM_1->    HX_SH:C  ST_HP1 ->STM_HP1
           ST_HP2 ->STM_HP2
           HX_RH:C  ST_IP ->STM_IP
           ST_LP1 ->STM_LP1
           ST_LP2 ->STM_LP2  ST_LP3 ->STM_DUM
           SC_1
STM_HP1->   FH_HP1:H <-STM_DUM
STM_HP2->   FH_HP2:H <-STM_HP1
STM_LP1->   FH_LP1:H <-STM_HP2
STM_LP2->   FH_LP2:H <-STM_LP1
STM_1->     MX_SC <-STM_LP2
           PUMP_SC HX_ECON:C  FH_LP2:C  FH_LP1:C
           DEAR_1 <-STM_IP
           PUMP_FW FH_HP2:C  FH_HP1:C
LIQ_1->     MX_FW <-STM_1
           PUMP_BFP HX_BOIL:C  IN_H2O:CYCL

```

We start with the liquid-water (actually two-phase) flow into the SD_1 model and then follow the STM_1 flow from the SD_1 model through the turbine train to the exit of the SC_1 model. At this point, it becomes necessary to mix this STM_1 flow with another, as yet unknown, flow. However, it is possible to go to the hot entry of the first high-pressure feedwater heater (FH_HP1) and continue processing the flows through the hot entries of each feedwater heater. The second flow of each of these feedwater entries is the cascaded flow from the previous high-pressure heater. The first high-pressure heater, not having such a cascaded flow, is fed a dummy steam flow. This dummy flow must be created like any other flow, even though its mass flow rate will be set to zero. The last turbine stage does not have any extraction flow (STM_DUM leads nowhere), so this flow may also be used for this first high-pressure feedwater flow. If the lower-pressure turbine stage had made use of this extraction flow, then a flow initiator would have been required to generate the STM_DUM flow.

By calling these hot-side feedwater-heater entries, the flow from the steam condenser (STM_1) can now be mixed with a known flow (STM_LP2). The STM_1 flow can then be processed through the pumps, cold-side entries to the feedwater heaters, and deaerator and finally mixed with the downcomer flow (LIQ_1) from the SD_1 model. This LIQ_1 flow is then followed through the boiler feed pump and the boiler and back to the liquid-flow initiator.

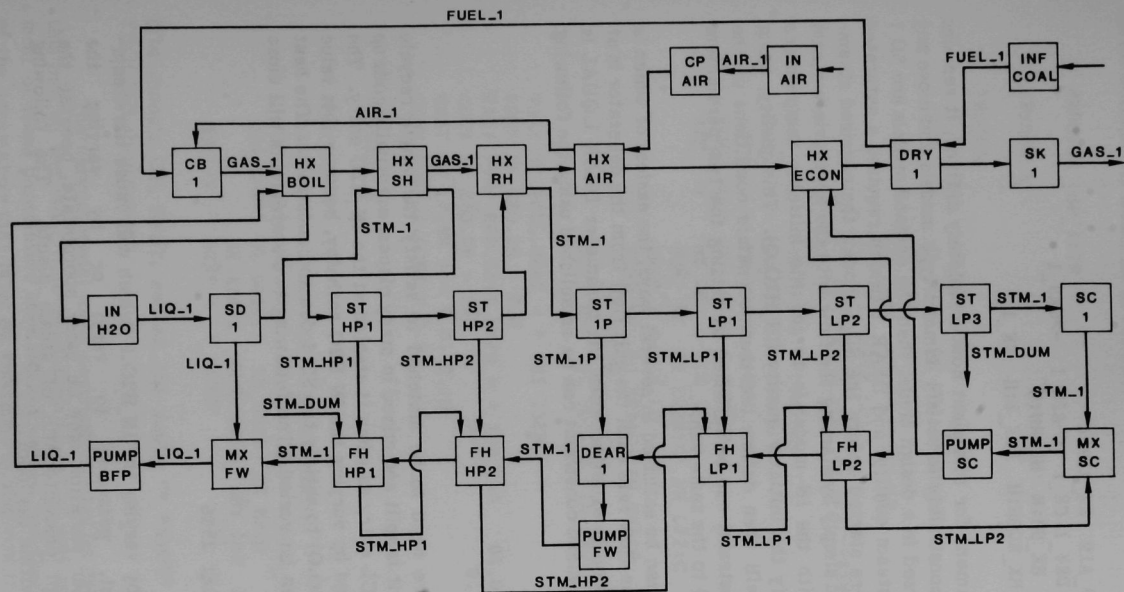


FIGURE 14 Fossil/Steam Power Plant

The PROCESS statements for the AIR₁ and GAS₁ flows are easily developed as follows:

```
PROCESS
AIR_1->  IN_AIR CP_AIR HX_AIR:C
FUEL_1->  INF_COAL DRY_1 CB_1 <-AIR_1 ->GAS_1
GAS_1->  HX_BOIL:H HX_SH:H HX_RH:H
          HX_AIR:H HX_ECON:H DRY_1:H SK_1
```

With the PROCESS statement for this plant now completely defined, it remains to add any necessary subsystem constraints to satisfy demand-type models and close any loops. If the plant is being analyzed in a design mode, then the demand models are SD₁ (which requires a specific inlet steam quality) and DEAR₁ (which requires a saturated exit flow). If the LIQ₁ flow were started before the SD₁ model, the required steam-drum constraint could be satisfied simply by starting the flow with the appropriate steam quality. One of the options with the IN model is to set the initiator temperature (IN_H2O.T) to zero and to specify the initiator quality (IN_H2O.Q). This quality, along with the pressure (IN_H2O.P), will then define the thermodynamic conditions of the initiated flow. Thus, the SD constraint can be satisfied by setting the two parameters SD₁.STEAM_QUAL and IN_H2O.Q to the same value.

The deaerator constraint can be satisfied in several ways, the easiest of which is simply to vary the ST_IP extraction flow rate until the exit flow from the deaerator is at saturation conditions. This condition is satisfied when the parameter DEAR₁.QUAL is equal to zero. Thus, this model demand constraint can be established using the following statements:

```
VARY ST_IP.SR = * 0.01 0.20
CONS DEAR_1.PARM.QUAL=0.0
```

Only enthalpy and pressure need to be matched to satisfy the single recycle loop. The mass flow, even though it is split and mixed in many places, eventually ends up being the same at the IN_H2O:CYCL entry as when it started at the IN_H2O entry. The enthalpy closure cannot be satisfied by varying the IN_H2O enthalpy, because this value is determined by specifying (IN_H2O.Q) to satisfy the SD₁ demand constraint. The heat load on the HX_BOIL, however, can be varied. The following two statements will close the enthalpy.

```
VARY HX_BOIL.HEAT = * 1E1 25E6
CONS IN_H2O.DH = 0.0
```

The pressure can be closed either by varying the IN_H2O.P (which will cause the steam-drum, and hence turbine-inlet, pressures to vary) or by varying the PUMP_BFP.EXIT_PRES. This latter variation may be more appropriate, because the drum pressure is usually a design characteristic of steam plants. The following statements will close the pressure within the loop:

```
VARY PUMP_BFP.EXIT_PRES = * 100 200
CONS IN_H2O.DP = 0.0
```

All three sets of VARY-CONS statements must be placed within a subsystem encompassing the entire water/steam flow path. Thus, the complete input (excluding the DATA statement) would be as follows:

```

PROCESS  GP_1:IN

SYSBEG  A

PROCESS
  LIQ_1->  IN_H2O  SD_1 ->STM_1
  STM_1->   HX_SH:C  ST_HP1 ->STM_HP1
           ST_HP2 ->STM_HP2
           HX_RH:C  ST_IP ->STM_IP
           ST_LP1 ->STM_LP1
           ST_LP2 ->STM_LP2  ST_LP3 ->STM_DUM
           SC_1
  STM_HP1-> FH_HP1:H <-STM_DUM
  STM_HP2-> FH_HP2:H <-STM_HP1
  STM_LP1-> FH_LP1:H <-STM_HP2
  STM_LP2-> FH_LP2:H <-STM_LP1
  STM_1->   MX_SC <-STM_LP2
           PUMP_SC HX_ECON:C  FH_LP2:C  FH_LP1:C
           DEAR_1 <-STM_IP
           PUMP_FW FH_HP2:C  FH_HP1:C
  LIQ_1->   MX_FW <-STM_1
           PUMP_BFP HX_BOIL:C  IN_H2O:CYCL

VARY HX_BOIL.HEAT = * 1E1 25E6
CONS IN_H2O.DH = 0.0
VARY PUMP BFP.EXIT_PRES = * 100 200
CONS IN_H2O.DP = 0.0
VARY ST_IP.SR = * 0.01 0.20
CONS DEAR_1.CONS = 0.0

SYSEND  A

PROCESS
  AIR_1->   IN_AIR  CP_AIR  HX_AIR:C
  FUEL_1->  INF_COAL  DRY_1  CB_1 <-AIR_1 ->GAS_1
  GAS_1->   HX_BOIL:H  HX_SH:H  HX_RH:H
           HX_AIR:H  HX_ECON:H  DRY_1:H  SK_1
  NULL->    SYST_1  *_*:OUT

```

(The models GP, SYST, and all the model " *_*:OUT" entries were also added to the PROCESS statements.)

The various model input parameters must be defined using the DATA statement; this can be the most time-consuming aspect of setting up the SALT input. Of course, most of these parameters have default values, and these values may be sufficient; many of the parameters may later be varied by adding constraints to the system. In any case, some set of values is required for each of the model input parameters.

A typical set of parameters for the two IN models might be as follows:

```
IN_AIR.PARM .ID='GAS'; .T=298.15; .P=1.0; .M=14.0;
             .XN2=0.78; .XO2=0.22; .XH2O=0.01;
IN_H2O.PARM .ID='H2O'; .T=0.0; .P=180.0; .M=75.0;
             .Q=0.20;
```

Here, for the AIR flow initiator, the flow ID was defined as "GAS," which implies that the GAS properties procedure is to be used to define the thermodynamic behavior of this flow throughout the system. When ID is equal to GAS, one must define the flow's constituents in terms of its molar fractions by species. In this case, we took the AIR flow to be 78% nitrogen, 21% oxygen, and 1% water vapor. (If desired, argon and carbon dioxide could also have been included.) For the steam/water flow (specifying the ID as "H2O"), the water properties code will be used to define the thermodynamic behavior. The temperature, pressure, and mass flow rate must also be specified for both flows. For the steam/water flow, the condition $T = 0$ implies that the flow is saturated, and its temperature will be calculated using the specified pressure. In this case, the flow's quality must be set in order to define the thermodynamic state of the flow uniquely. This quality is to be the same as that of the drums; thus, for the steam-drum model, the following must also be specified:

```
SD_1.PARM .STEAM_QUAL=0.20;
```

For the INF model initiating the fuel flow, the parameters for a typical coal might be

```
INF_COAL.PARM .M=2.0; .C=0.5213; .H=0.060; .O=0.3152;
               .N=0.0079; .S=0.0085; .H2O=0.227;
               .ASH=0.0871; .HHV=20.743E6;
```

while for the other components in the AIR and GAS flow paths, a typical set of parameters might be

```
CP_AIR.PARM .EXIT_PRES=1.15; .EFFICIENCY=0.88;
CB_1.PARM .ASH_DET=0.0;
HX_BOIL.PARM .HEAT=12E6;
HX_SH.PARM .T_SET(2)=811;
HX_RH.PARM .T_SET(2)=811;
HX_AIR.PARM .T_SET(2)=500;
HX_ECON.PARM .HEAT=1E5;
DRY_1.PARM .H2O_DET=0.05;
```

Many other parameters exist for these models; failure to specify them implies that their default values will be used. For the HX models, these default values will be used to determine heat-transfer surface areas, so these outputs may not be accurate in this example. On the other hand, the results for a simple heat and mass balance will not be affected. The T_SET parameter was used for three of the HX models to define the outlet temperature of the cold flow. This parameter should be used carefully, because the T_SET element (1 or 2) must correspond to the entry ("H" or "C") that is used first in the configuration.

For the turbine/feedwater train, a typical set of inputs might be:

```
ST_HP1.PARM .EXIT_PRES=100.; .EFFICIENCY=0.84; .SR=0.10;
ST_HP2.PARM .EXIT_PRES=50.; .EFFICIENCY=0.84; .SR=0.10;
ST_IP.PARM .EXIT_PRES=15.; .EFFICIENCY=0.86; .SR=0.07;
ST_LP1.PARM .EXIT_PRES=5.; .EFFICIENCY=0.87; .SR=0.05;
ST_LP2.PARM .EXIT_PRES=1.; .EFFICIENCY=0.87; .SR=0.05;
ST_LP3.PARM .EXIT_PRES=0.066; .EFFICIENCY=0.87; .SR=0.0;
SC_1.PARM .EXIT_PRES=0.066;
PUMP_SC.PARM .EXIT_PRES=15.0; .EFFICIENCY=0.90;
PUMP_FW.PARM .EXIT_PRES=180.0; .EFFICIENCY=0.90;
PUMP_BFP.PARM .EXIT_PRES=190.0; .EFFICIENCY=0.90;
```

Here, all parameters for the feedwater heaters were taken at their default values. For three of the model parameters -- HX_BOIL.HEAT, PUMP_BFP.EXIT_PRES, and ST_IP.SR -- the values assigned in the DATA statement will be used only as initial guesses in the VARY statements. The parametric values assigned must lie within the lower and upper bounds used in the VARY statements.

This completes all the input necessary to do a system run for this problem. However, additional user-imposed constraints (those that cannot be met simply by setting values of the model input parameters) may be added to the system before the run is made. For instance, it might be desired to determine the inlet liquid/steam mass flow rate to the drum such that the gas temperature from the boiler is 1650 K. This constraint would be accomplished by adding

```
VARY IN H2O.M = * 1 50
CONS HX_BOIL.FLH.TEMP = 1650
```

to the appropriate subsystem, which could be an additional subsystem encompassing the original one around the steam/water path and including the flow paths of AIR, FUEL, and GAS (at least, down through the boiler model). Alternatively, these statements could be included in the original subsystem to satisfy the model demand constraints and recycle loop constraint by enlarging that subsystem to include the additional AIR, FUEL, and GAS paths.

Appendix D shows the entire SALT input for this problem, as well as the resulting outputs. All of the outputs, other than those generated by the mathematical utilities (i.e., equation solvers and optimizers), are actually generated by the models themselves. Thus, if the outputs are unsatisfactory, the user need only change the models, not the SALT code. Some attempt has been made to keep the model outputs reasonably similar in appearance, although differing amounts of output data are generated by different models. Usually, only the model parameters are printed out for each model output entry; the flows are printed out by the SYST model's output entry. Because the flow tables, power summary table, etc. are printed by the SYST model, it is best not to have other models appear after that model in the SALT input. The model output entries are called in the order of their appearance within the PROCESS statements. Because SYST uses the substructures of the other models in calculating power, any model with a POWER substructure should come before the SYST model.

The table of output by flows printed by the SYST model actually will be the values of the flows saved in the models' substructures of FLH, FLC, etc. These values usually represent the exit flow conditions, except where the flow is an input flow to the model. For example, the second flow to the MX model (as printed out in the flow-table summary) would be the input to that model and would usually be the last model using that flow.

Two other types of output are generated by the SALT code. The first is the output generated by the mathematical procedures, and the second is a summary of the success or failure of such procedures. The first type of data, generated while the code is attempting to complete the subsystem task, is printed before the model output entries are called (unless such entries are specified within the subsystem loop).

Varying amounts of information may be generated, depending on the value of the PRINT switch specified in the SWITCH statement. If PRINT is equal to zero, no output is generated. If PRINT is equal to one, the output generated consists of the name of the subsystem; the iteration number being performed (denoted "N="); the objective function or the sum of the squares of the constraint violations, depending on whether or not the problem is an optimization (denoted "F="); the values of the independent variables -- i.e., the VARY parameters (denoted as "X="); and the values of the constraint violations (denoted "C="). In an optimization, the constraints are reordered from their order of appearance within the SALT input so that the equality constraints appear first, followed by any inequality constraints.

For larger values of PRINT, additional output will be generated to help identify convergence problems. This additional output (to be discussed later) requires some familiarity with the mathematical procedures.

The success/failure data are used to summarize the final iteration of all subsystems. This information, which is printed whenever the system problem calls the equation solver or optimizer (regardless of the value of the PRINT switch), is always the last output generated by the SALT code. The data include the subsystem name and a statement of the type of termination (normal or otherwise), the final objective-function value or the sum of the squares of the constraint violations, the variable values and their names, the constraint violations, and the constraints themselves.

5.3.2 Open-Cycle Magnetohydrodynamic Power Plant

Figure 15 shows a simple open-cycle MHD plant; the system configuration is basically the same as that shown in Fig. 14, but with the addition of an MHD topping cycle consisting of a nozzle (NZ), an MHD channel (MG), and a diffuser (DF). In order to provide a sufficiently high combustion temperature, the inlet air flow has been mixed with an additional oxygen flow before being preheated and compressed. The compression (to 6 atm) is necessary for the MHD channel's operation. By tracing through the AIR, FUEL, GAS, and STM flows, one may write the PROCESS statement for the configuration as follows:

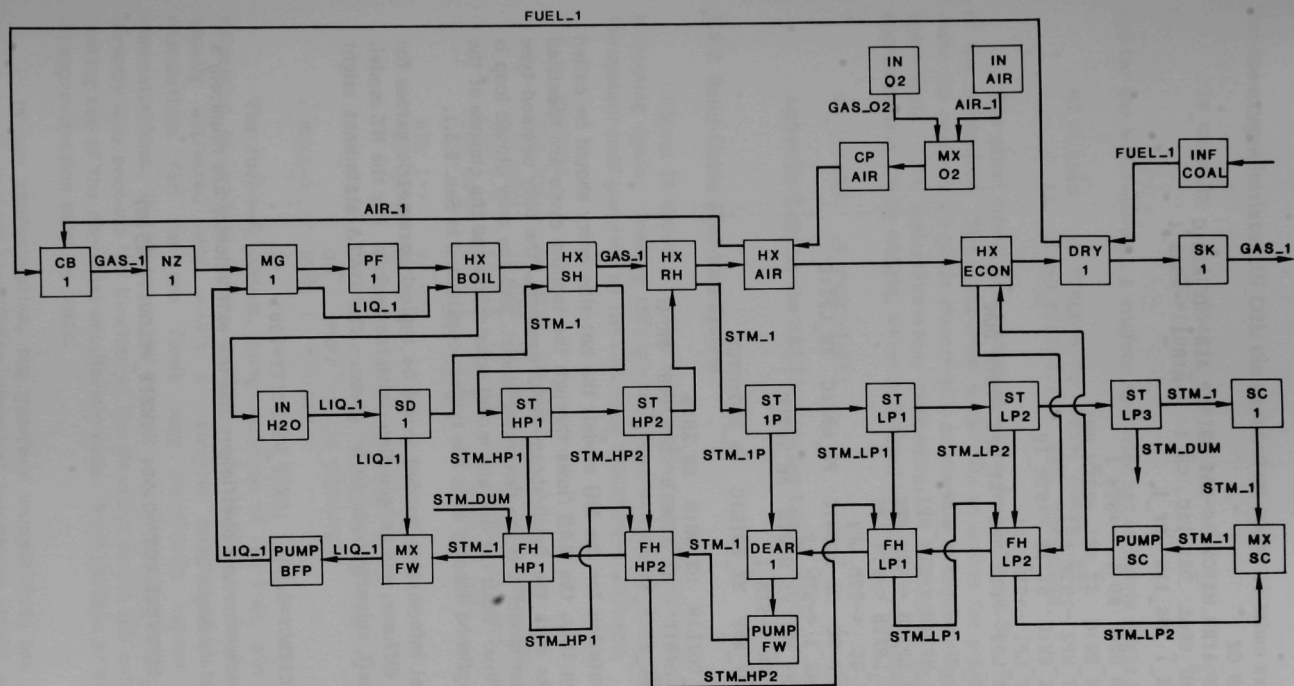


FIGURE 15 Open-Cycle Magnetohydrodynamic Power Plant

```

PROCESS
  GAS_O2->  IN_O2
  AIR_1->    IN_AIR  MX_O2  <-GAS_O2  CP_AIR  HX_AIR:C
  FUEL_1->   INF_COAL  DRY_1:C  CB_1  <-AIR_1  ->GAS_1
  GAS_1->    NZ_1  MG_1:H  DF_1

  LIQ_1->    IN_H2O  SD_1  ->STM_1
  STM_1->    HX_SH:C  ST_HP1  ->STM_HP1
               ST_HP2  ->STM_HP2
               HX_RH:C  ST_IP  ->STM_IP
               ST_LP1  ->STM_LP1
               ST_LP2  ->STM_LP2  ST_LP3  ->STM_DUM
               SC_1

  STM_HP1->   FH_HP1:H  <-STM_DUM
  STM_HP2->   FH_HP2:H  <-STM_HP1
  STM_LP1->   FH_LP1:H  <-STM_HP2
  STM_LP2->   FH_LP2:H  <-STM_LP1
  STM_1->     MX_SC  <-STM_LP2
               PUMP_SC  HX_ECON:C  FH_LP2:C  FH_LP1:C
               DEAR_1  <-STM_IP
               PUMP_FW  FH_HP2:C  FH_HP1:C
  LIQ_1->     MX_FW  <-STM_1
               PUMP_BFP  HX_BOIL:C  IN_H2O:CYCL

  GAS_1->     HX_BOIL:H  HX_SH:H  HX_RH:H
               HX_AIR:H  HX_ECON:H  DRY_1:H  SK_1
  NULL->      SYST_1  * *:OUT

```

According to the documentation for the MG model, the hot-side entry should be called before the cold-side entry; thus, the GAS flows through the topping cycle are specified before the STM flows. As with the fossil/steam-plant example, the only demand-type models are the steam-drum model and the deaerator model, and the only closed loop is that of the water/steam flow. Both the demand model constraints and the closure of the recycle loop can be accomplished exactly as in the system considered in Sec. 5.3.1.

The only additional information needed would be typical parametric values for the nozzle, MHD channel, diffuser, and oxygen-flow-initiator models. For the NZ model, only the efficiency and exit velocity need to be given. The DATA statement might include the following:

```
NZ_1.PARM .EFFICIENCY=0.90; .EXIT_VELOCITY=750;
```

For the diffuser, the pressure-recovery coefficient and the exit velocity are required; a typical set of these parameters might be

```
DF_1.PARM .PRES_RECOVERY_COEF=0.50; .EXIT_VELOCITY=25.0;
```

For the oxygen initiator (IN_O2), the parameters might be taken as

```
IN_O2.PARM      .ID='GAS'; .T=298.15; .P=1.0; .M=2.0;
                  .XO2=1.0;
```

and for the MHD channel, a minimal specified set of inputs might be

```
MG_1.PARM      .B_FIELD=6.0; .EXIT_PRES=0.85;
                  .LOAD_FACT=0.7; .WALL_TEMP=1850.;
```

The other model parameters could be treated as in Sec. 5.3.1, with the exception of the combustor. The combustor model has an option for potassium seed injection to make the combustion gases electrically conductive (a requirement for the MHD channel's operation). The potassium-atom concentration needed in the combustion gases is approximately 1% by weight, which is defined using the combustor parameter (K_FRAC):

```
CB_1.PARM      .K_FRAC=0.01;
```

Appendix E shows the SALT output for this problem.

5.3.3 Solid-Oxide Fuel-Cell System

Figure 16 shows a typical solid-oxide fuel-cell plant with a simple steam-turbine bottoming cycle. Tracing through the flows as labeled in the figure, we can represent the system configuration by the following PROCESS statement:

```
PROCESS
STM_MIX->  IN_MSTM CP_STM
GAS_AN->   IN_GAS CP_GAS HX_1:C MX_STM <-STM_MIX HX_A:C
AIR_1->    IN_AIR CP_AIR1 HT_INTER CP_AIR2 HX_C:C
GAS_AN->   SOFC_1 <-AIR_1 MX_BURN <-AIR_1
            SP_BURN ->AIR_1
AIR_1->    HX_C:H
GAS_AN->   HX_A:H MX_AIR <-AIR_1
            HX_FB:H GT_1 HX_ST:H SK_1
STM_1->    IN_STM HX_ST:C HX_FB:C HX_1:H ST_1 ->STM_DUM
            SC_1 PUMP_SC IN_STM:CYCL
NULL->     SYST_1 *_*:OUT
```

The fuel-cell system, using methane as the fuel, will require methane/steam reforming to produce carbon monoxide and hydrogen. Such reforming processes are usually activated only within a catalytic environment, so chemical-equilibrium calculations for the gas flows may incorrectly represent the flow's species concentrations. In this case, the correct way to handle the gas compositions is to take directly into account the kinetics of the situation inside the component models, without making use of the chemical-equilibrium code. Most models, at present, do not do this, so an approximation must be made.

In our approximation, the gas-flow compositions are frozen except at those locations where near-equilibrium chemical conditions would prevail. These locations

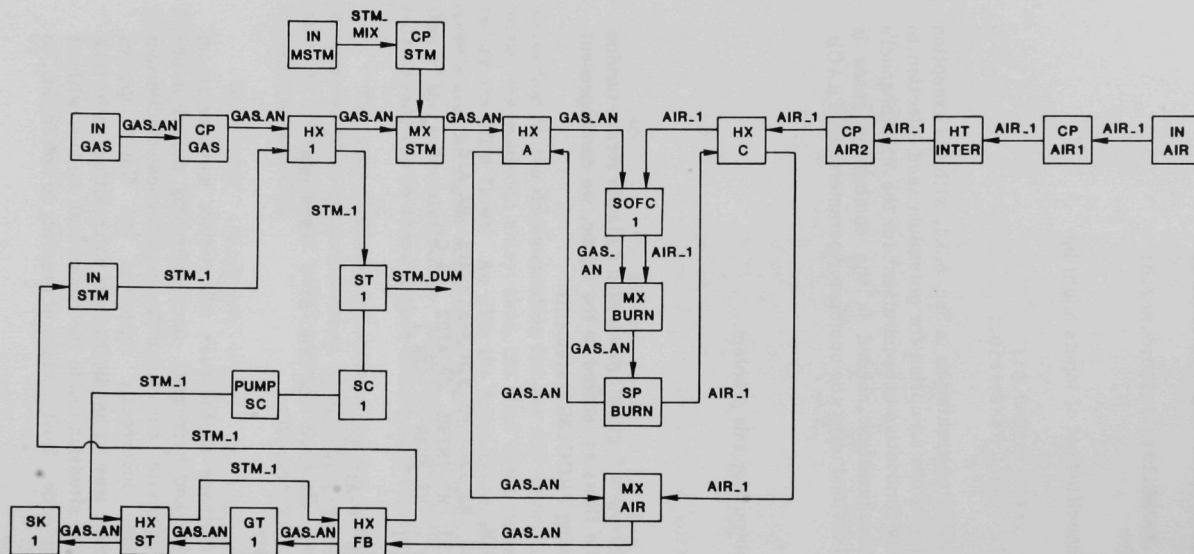


FIGURE 16 Solid-Oxide Fuel-Cell System

might be downstream of a reformer, or they might follow a burning process or some other high-temperature process. In order to freeze and unfreeze the gas-species concentrations, a parameter within the gas properties code (denoted GASFRZ) should be set to one (for freezing) or zero (for unfreezing). This parameter is declared within the GP model's interface and can be set using the PLI key word.

In order to include nonequilibrium effects within the system modeling, the PROCESS statement can be split into several statements and the PLI key word used along with GASFRZ to effectively turn chemical equilibrium calculations on and off along the flow path. Initially, all the gas compositions may be frozen and then, right before the SOFC model, be unfrozen. Technically, the AIR flow would not need to be frozen; however, it consists only of nitrogen and oxygen, so very little error will occur if it is frozen also. The flows could be frozen following the mixer that represents the burner. Evaluation of the thermodynamic properties of a flow is considerably faster with frozen compositions, because the chemical equilibrium calculations are bypassed. With the PLI statements included, the system configuration would become the following:

```

PLI GASFRZ=1;
PROCESS
  STM_MIX->   IN_MSTM  CP_STM
  GAS_AN->    IN_GAS   CP_GAS  HX_1:C  MX_STM <-STM_MIX HX_A:C
  AIR_1->     IN_AIR   CP_AIR1  HT_INTER CP_AIR2  HX_C:C
PLI GASFRZ=0;
PROCESS
  GAS_AN->    AIR_1->  SOFC 1
  GAS_AN->    MX_BURN  <-AIR_1
                  SP_BURN ->AIR_1
PLI GASFRZ=1;
PROCESS
  AIR_1->     HX_C:H
  GAS_AN->    HX_A:H  MX_AIR <-AIR_1
                  HX_FB:H  GT_1  HX_ST:H  SK_1
  STM_1->     IN_STM  HX_ST:C  HX_FB:C  HX_1:H  ST_1 ->STM_DUM
                  SC_1  PUMP_SC  IN_STM:CYCL
  NULL->      SYST_1  *_*:OUT

```

There are no demand model constraints, so the closure of the single steam-recycle loop now must be considered. The enthalpy and pressure need to be closed. Closure of the pressure is achieved by setting IN_STM.P equal to PUMP_SC.EXIT_PRES. The enthalpy closure could be established in several ways; here, we vary the steam mass flow rate until the enthalpies match. A subsystem (consisting of the steam loop) will be iterated over the following statements:

```

VARY  IN_STM.M = * 1 10
CONS  IN_STM.DH = 0.0

```

No fuel-flow or stream combustor model was used in representing this fuel-cell system; the methane flow was initiated using an IN model. Because no model that generates values of POWER.INPUT would have been used (the IN model would generate such a value, but only when called with the INCYCL entry), the efficiency calculations performed by the SYST model would not be correct. To overcome this difficulty,

the IN_GAS.POWER.INPUT parameter is assigned a value using the higher heating value of the methane and a PL/I statement. This PL/I statement is placed within the configurational statements before the SYST model (which makes use of this parameter) is called:

```
PLI IN_GAS.POWER.INPUT=IN_GAS.PARM.M*55.5E6;
```

Other changes to the power structures can be introduced at the same time. For example, the fuel-cell model does not include an inverter efficiency, so one might adjust its produced power to reflect such an inverter loss as follows:

```
PLI SOFC_1.POWER.PRODUCED=0.96*SOFC_1.POWER.PRODUCED;
```

Taking into account all of these considerations, we obtain the complete SALT input (excluding the DATA statement) for this fuel-cell system as follows:

```
PROCESS GP_1:IN
PLI GASFRZ=1;
PROCESS
  STM_MIX-> IN_MSTM CP_STM
  GAS_AN-> IN_GAS CP_GAS HX_1:C MX_STM <-STM_MIX HX_A:C
  AIR_1-> IN_AIR CP_AIR1 HT_INTER CP_AIR2 HX_C:C
PLI GASFRZ=0;
PROCESS
  GAS_AN-> SOFC_1 <-AIR_1 MX_BURN <-AIR_1
  SP_BURN ->AIR_1
PLI GASFRZ=1;
PROCESS
  AIR_1-> HX_C:H
  GAS_AN-> HX_A:H MX_AIR <-AIR_1
  HX_FB:H GT_1 HX_ST:H SK_1
SYSBEG A
VARY IN_STM.PARM.M = * 1.0 10.0
CONS IN_STM.DH = 0.0
PROCESS
  STM_1-> IN_STM HX_ST:C HX_FB:C HX_1:H ST_1 ->STM_DUM
  SC_1 PUMP_SC IN_STM:CYCL
SYSEND A
PLI IN_GAS.POWER.INPUT = IN_GAS.PARM.M*55.5E6;
  SOFC_1.POWER.PRODUCED= 0.96*SOFC_1.POWER.PRODUCED;
PROCESS
  NULL-> SYST_1 *_*:OUT
```

A typical DATA statement for this problem might be the following:

```
DATA
  IN_GAS.PARM .ID='GAS'; .T=298.15; .P=1.0; .M=1.0;
  .XCH4=1.0;
  IN_STM.PARM .ID='H2O'; .T=823.0; .P=150.; .M=5.0;
  IN_MSTM.PARM .ID='H2O'; .T=298.15; .P=1.0; .M=1.60;
  IN_AIR.PARM .ID='GAS'; .T=298.15; .P=1.0; .M=30.0;
```

```

.XO2=0.21; .XN2=0.79;
HX_1.PARM .T_SET(2)=573.0;
HX_A.PARM .T_SET(2)=1073.0;
HX_C.PARM .T_SET(2)=1073.0;
HX_FB.PARM .T_SET(1)=800.0;
HX_ST.PARM .T_SET(1)=353.0;
CP_AIR1.PARM .EXIT_PRES=3.5; .EFFICIENCY=0.85;
CP_AIR2.PARM .EXIT_PRES=12.0; .EFFICIENCY=0.85;
HT_INTER.PARM .T_SET=318.0;
CP_GAS.PARM .EXIT_PRES=13.0; .EFFICIENCY=0.85;
CP_STM.PARM .EXIT_PRES=12.0; .EFFICIENCY=0.85;
SOFC_1.PARM .CELL_CURRENT=1.5686E5; .NO_OF_CELLS=230;
.CELL_TEMP=1273.;
SP_BURN.PARM .SPLIT_RATIO=0.7;
ST_1.PARM .EXIT_PRES=0.180; .EFFICIENCY=0.82;
SC_1.PARM .EXIT_PRES=0.180;
GT_1.PARM .EXIT_PRES=1.0; .EFFICIENCY=0.87;
PUMP_SC.PARM .EXIT_PRES=150.0;
SYST_1.PARM .POWER_HEAD_PTR=POWER_HEAD_PTR;
.FLOW_HEAD_PTR=FLOW_HEAD_PTR;

```

Many other parameters for these models exist. For example, if heat-exchanger surface-area calculations were important, additional parameters would have to be specified.

Appendix F Shows the SALT output for this problem.

5.3.4 Liquid-Metal Magnetohydrodynamic System

Figure 17 shows a liquid-metal MHD system employing a Brayton-cycle helium gas loop and a sodium-metal liquid loop. The models used for liquid-metal MHD systems treat the two-phase, two-component flows as separately specified flows. Thus, even though only a single two-component flow enters a component, each flow component is specified as a different flow.

The two-phase, two-component flow entering the liquid-metal MHD generator model (MMHD) is initiated by the two inlet models, one for the helium component (IN_GAS) and one for the liquid-sodium component (IN_LIQ). These two flows pass through the generator model, a two-phase nozzle model (TPNZ), and a liquid/gas separator model (SEPR). This last model, which separates the two flows, generates two additional flows representing liquid and gaseous carry-overs in the gaseous and liquid flows, respectively. These carry-over flows would have to be dealt with separately in an actual system, but for simplicity we will specify their flow rates as zero and assume that the separator separates the two phases perfectly. The gaseous flow from the separator is then passed through the hot side of a regenerator modeled as a heat exchanger (HX_REG), a radiator modeled as a heater with a negative heat load (HT_COOL), a gas compressor (CP_GAS), and (finally) the cold side of the regenerator. The liquid flow from the separator passes through a liquid-metal diffuser (MDIF_1), a heater (HT_LIQ), and a liquid-metal nozzle (MNOZ_1). The gaseous and liquid flows are then mixed in the TPMX_1 model and cycled back into their respective initiator models. The whole configuration is represented by the following PROCESS statement:

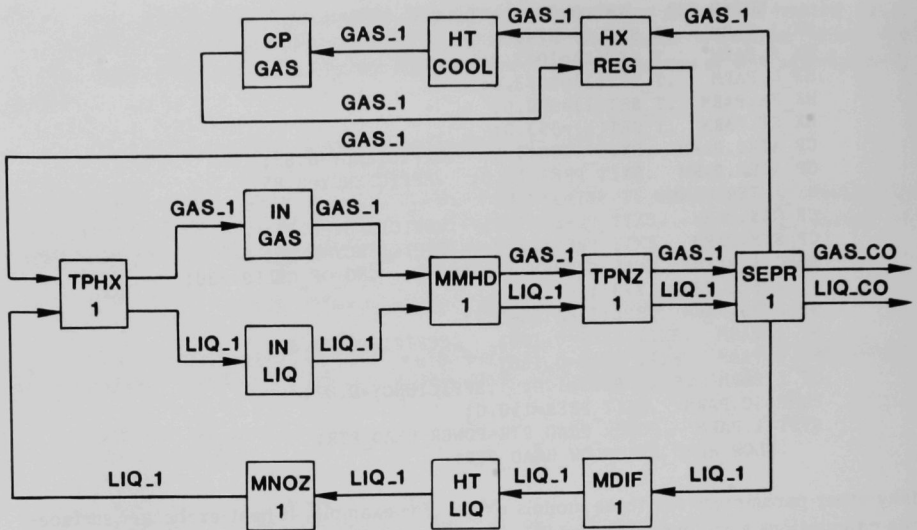


FIGURE 17 Liquid-Metal Magnetohydrodynamic System

PROCESS

```

GAS_1-> IN_GAS
LIQ_1-> IN_LIQ
GAS_1-> LIQ_1-> MMHD_1 TPNZ_1 SEPR_1 ->GAS_CO ->LIQ_CO
LIQ_1-> MDIF_1 HT_LIQ MNOZ_1
GAS_1-> HX_REG:H HT_COOL CP_GAS HX_REG:C
GAS_1-> LIQ_1-> TPMX_1
GAS_1-> IN_GAS:CYCL
LIQ_1-> IN_LIQ:CYCL

```

This system has one demand-type model, the TPMX, which requires that the two inlet flows have the same pressure. This model assigns the difference in the inlet flow pressures to the parameter PRES_DIFF_IN, which should be constrained to equal zero by suitably varying some upstream condition in either the liquid or gaseous flows. For example, the gas-compressor exit pressure could be varied to satisfy this constraint:

```

VARY CP_GAS.EXIT_PRES = * 40 60
CONS TPMX_1.PRES_DIFF_IN = 0.0

```

The system also has two recycle loops (both the gaseous and liquid flows close on themselves). It is important to make the mass flow rates, temperatures, and pressures match at the initiator models. In liquid-metal systems, it is also important to make the velocities match; a large portion of the energy in liquid-metal flows is carried in the velocity heads.

We have assumed perfect flow separation within the separator model, so the mass flow rates will match automatically for any flow rates chosen. By choosing the recycle point downstream of the TPMX model, we can close both loops by closing only one; this situation exists because the TPMX model has common values for the pressures, temperatures, and velocities of its exiting flows. The flow velocities out of the TPMX are equal to the liquid velocities into the mixer; therefore, if the exit velocity of the liquid-metal nozzle is set equal to the initiator velocity, the velocities for both flows will be closed.

For the temperature closure, one might vary the heat input to the liquid flow until the temperature difference in the liquid initiator went to zero, as follows:

```
VARY HT_LIQ.HEAT = * 1E2 5E6
CONS IN_LIQ.DT = 0.0
```

This procedure also would force IN_GAS.DT to equal zero, provided the gaseous and liquid flows were initiated at the same temperature. Alternatively, one might vary the liquid initiator temperature until IN_LIQ.DT equaled zero. (In this case, the IN_GAS.T parameter should be set equal to the IN_LIQ.T value by means of a PLI statement.)

For the pressure closure, some condition upstream of the IN_LIQ:CYCL entry must be varied until IN_LIQ.DP is equal to zero. The two-phase-nozzle model's exit pressure might be used for this task, as follows:

```
VARY TPNZ_1.EXIT_PRES = * 3 47
CONS IN_LIQ.DP = 0.0
```

These statements would also close the gas-side pressure, provided that the gaseous and liquid flows were initiated at the same pressure. As with the closure of any recycle loop, some consideration of the variables used to close the loop is recommended. For example, varying the MMHD.EXIT_PRES would not achieve closure, because the pressure downstream of the TPNZ would be the same (i.e., TPNZ_1.EXIT_PRES) no matter what pressure was chosen at the generator's exit.

The complete SALT input statements (excluding the DATA statement) for this problem can be written as follows:

```
PROCESS GP_1:IN
SYSBEG A
PROCESS
  GAS_1-> IN_GAS
  LIQ_1-> IN_LIQ
  GAS_1-> LIQ_1-> MMHD_1 TPNZ_1 SEPR_1 ->GAS_CO ->LIQ_CO
  LIQ_1-> MDIF_1 HT_LIQ MNOZ_1
  GAS_1-> HX_REG:H HT_COOL CP_GAS HX_REG:C
  GAS_1-> LIQ_1-> TPMX_1
  GAS_1-> IN_GAS:CYCL
  LIQ_1-> IN_LIQ:CYCL
VARY HT_LIQ.HEAT = * 1E2 5E6
CONS IN_LIQ.DT = 0.0
```

```

VARY CP_GAS.EXIT_PRES = * 40 60
CONS TPMX_1.PRES_DIFF IN = 0.0
VARY TPNZ_1.EXIT_PRES = * 3 47
CONS IN_LIQ.DP = 0.0
SYSEND A
PROCESS
  NULL-> SYST_1 *_*:OUT

```

A typical DATA statement might be taken as:

```

DATA
  IN_GAS.PARM .ID='THR-HE'; .T=867; .P=50.0; .V=25; .M=1.0;
  IN_LIQ.PARM .ID='THR-NA'; .T=867; .P=50.0; .V=25; .M=100.;
  MMHD_1.PARM .EXIT_PRES= 24.0; .EFFICIENCY=0.80;
  TPNZ_1.PARM .EXIT_PRES=20.0; .EFFICIENCY=0.90;
  SEPR_1.PARM .VELOCITY_HEAD_RATIO=0.90;
  HX_REG.PARM .HEAT=6E4;
  HT_COOL.PARM .HEAT=-6.5E5;
  HT_LIQ.PARM .HEAT=1.0E6;
  CP_GAS.PARM .EXIT_PRES=50.0 .EFFICIENCY=0.90;
  MDIF_1.PARM .EXIT_VELOCITY=15.0; .EFFICIENCY=0.90;
  MNOZ_1.PARM .EXIT_VELOCITY=25.0; .EFFICIENCY=0.90;
  TPMX_1.PARM .PRES_DROP=0.0;
  SYST_1.PARM .POWER_HEAD_PTR=POWER_HEAD_PTR;
               .FLOW_HEAD_PTR=FLOW_HEAD_PTR;

```

The gas and liquid initiator models have been assigned the same temperature, pressure, and velocity values; also, the MNOZ_1 exit velocity has been assigned the same value as the flow-initiator velocities, in accordance with the discussion above. The other model parameters have been assigned typical values. The input parameters for the liquid-metal diffuser and nozzle models include the exit velocity rather than the exit pressure.

For this problem, the input statements listed above are all that are really required. However, we can elect to impose some additional constraints on the inlet and exit void fractions of the channel and also to maximize the system efficiency. A typical set of inlet- and exit-channel void fractions might range from 0.55 to 0.85. Bearing in mind that the TPMX_1 exit void fraction is the channel's inlet void fraction, we can impose the void-fraction range over the channel by means of the following two inequality constraints:

```

CONS MMHD_1.VOID_FRACTION<0.85
CONS TPMX_1.VOID_FRACTION>0.55

```

The objective function is the system efficiency, which can be maximized by minimizing its negative:

```

MINI -SYST_1.EFFICIENCY

```

Additional parameters must now be varied to provide for the additional degrees of freedom needed for the optimization problem. These parameters might be any of the

additional model inputs, such as IN_GAS.M, IN_LIQ.M, MMHD_1.EXIT_PRES, or HT_REG.HEAT. The size of the system is not specified, but it can be somewhat fixed by keeping at least one of the mass-flow rates fixed (say IN_GAS.M = 1.0). The other parameters would generate three additional degrees of freedom. If the HT_REG.HEAT were to be varied, one might impose an additional constraint on the regenerator to prevent a pinch-point violation. For example, it might be appropriate to keep the exit temperature out of the cold side at least 20 K lower than the entering temperature on the hot side, where this last temperature is the same as the gas temperature leaving the SEPR_1 model:

```
CONS HX_REG.FLC.TEMP<SEPR_1.FLC1.TEMP-20.0
```

The complete SALT input statements for this problem, with these additional constraints and objective functions, would be as follows:

```
PROCESS GP_1:IN
SYSBEG A
PROCESS
  GAS_1-> IN_GAS
  LIQ_1-> IN_LIQ
  GAS_1-> LIQ_1-> MMHD_1 TPNZ_1 SEPR_1 ->GAS_CO ->LIQ_CO
  LIQ_1-> MDIF_1 HT_LIQ MNOZ_1
  GAS_1-> HX_REG:H HT_COOL CP_GAS HX_REG:C
  GAS_1-> LIQ_1-> TPMX_1
  GAS_1-> IN_GAS:CYCL
  LIQ_1-> IN_LIQ:CYCL
  NULL-> SYST_1
VARY HT_LIQ.HEAT = * 1E2 5E6 CONS IN_LIQ.DT = 0.0
VARY CP_GAS.EXIT_PRES = * 40 60 CONS TPMX_1.PRES_DIFF_IN = 0.0
VARY TPNZ_1.EXIT_PRES = * 3 47 CONS IN_LIQ.DP = 0.0
VARY HX_REG.HEAT = * 2E4 4E6
CONS HX_REG.FLC.TEMP>SEPR_1.FLC1.TEMP-20.0
VARY IN_LIQ.M = * 1.0 450 CONS MMHD_1.VOID_FRACTION < 0.85
VARY MMHD_1.EXIT_PRES = * 8 48 CONS TPMX_1.VOID_FRACTION > 0.55
MINI -SYST_1.EFFICIENCY
SYSEND A
PROCESS
  NULL-> * *:OUT
DATA
  IN_GAS.PARM .ID='HE'; .T=867; .P=50.0; .V=25; .M=1.0;
  IN_LIQ.PARM .ID='NA'; .T=867; .P=50.0; .V=25; .M=100.;
  MMHD_1.PARM .EXIT_PRES= 24.0; .EFFICIENCY=0.80;
  TPNZ_1.PARM .EXIT_PRES=20.0; .EFFICIENCY=0.90;
  SEPR_1.PARM .VELOCITY_HEAD_RATIO=0.90;
  HX_REG.PARM .HEAT=6E4;
  HT_COOL.PARM .HEAT=-6.5E5;
  HT_LIQ.PARM .HEAT=1.0E6;
  CP_GAS.PARM .EXIT_PRES=50.0 .EFFICIENCY=0.90;
  MDIF_1.PARM .EXIT_VELOCITY=15.0; .EFFICIENCY=0.90;
  MNOZ_1.PARM .EXIT_VELOCITY=25.0; .EFFICIENCY=0.90;
```

```
TPMX_1.PARM .PRES_DROP=0.0;  
SYST_1.PARM .POWER_HEAD_PTR=POWER_HEAD_PTR;  
           .FLOW_HEAD_PTR=FLOW_HEAD_PTR;
```

Appendix G shows the SALT output for this problem.

6 FAILURE CAUSES AND CURES

6.1 INTRODUCTION

Little was said in Chapter 5 about possible problems that might arise in connection with the examples presented there. In reality, it is very possible that the system problem that is set up will not converge to a solution. Of course, if no iterative tasks -- such as establishing constraints or performing optimizations -- are defined, then no system-level convergence problems will occur. However, even in these cases, some of the models may involve iterative processes that (depending on the model's input parameters and flows) could cause problems. Most of the existing models are sufficiently robust that this usually will not happen, given reasonable model inputs. In this chapter, we are more concerned with problems that result from system-level failures.

To consider all of the many different types of problems that can lead to a convergence failure would be impossible. Failures can be classified broadly as being due either to the inability of numerical procedures to handle the problem or to incorrect posing of the problem itself. Each of these two general types of failures can be further subdivided.

6.2 FAILURES DUE TO MATHEMATICAL PROBLEMS

If the physical problem is reasonably well posed (that is, there exists a solution to the problem that can in fact be found by the system and subsystem tasks that have been set up), then the failure probably results from some inadequacy in the mathematical procedures. All of the mathematical procedures have various types of return messages to indicate why they stopped. (One of these messages, of course, is that of "normal termination," indicating that a solution -- possibly one of many -- was found, at least within the accuracy specified by the user.) The reasons for termination are usually different for equality-constraining tasks and optimization tasks, and the two will be discussed separately.

6.2.1 Constraining Tasks

Subsystems that employ only equality constraints make use of an equation solver (SOLVG) that is a hybrid steepest-descent/quasi-Newtonian update technique. This technique, at present, has seven different terminating modes and four input parameters (other than the equations and variables and their bounds), which can be used to control the procedure. These four input parameters -- ACC, DEL, MAXIT, and PRINT -- can be assigned specific values by using the SALT code's SWITCH statement.

The ACC parameter represents the maximum value of the square root of the sum of the squares of the constraint violations (i.e., root-mean-square norm) that will be permitted at normal termination. The constraint violations are defined as the expression

on the left-hand side of the constraint equation minus that on the right-hand side. Thus, the constraint violation for

$$\text{CONS EXP1} = \text{EXP2}$$

is defined as $\text{EXP1} - \text{EXP2}$. The default value of ACC is taken as 0.1, a value that may seem somewhat large. However, for many system constraints the expressions represent enthalpies, powers, etc. that may be on the order of 10^6 . If a subsystem is set up with constraint violations within a much smaller range, then ACC should be set to a suitably small number.* It is important to consider the subsystem's constraints when defining a value of ACC.

For problems with one large constraint violation and one small constraint violation, it is usually enough to set the value of ACC necessary for reasonable convergence of the larger constraint. The reason for this is that, internally, the equation solver will scale all of the constraints so that they are considered approximately equally. As the larger constraint violations are driven smaller, the smaller constraint violations are driven even smaller. The internal scaling, which depends on the initial Jacobian of the set of equations defining the constraints, is not infallible.** It is always best, even with normal termination, to look at the individual constraint violations at the solution point. The SALT code will provide a summary of the variables and constraint violations for each subsystem task.

One failure mode of the equation solver involves simply hitting the maximum number of iterations specified by the MAXIT parameter, which has a default value of 40. For large problems, or problems that are extremely nonlinear, this value may be inadequate. Hitting the maximum number of iterations does not, in itself, indicate that the equation solver cannot solve the problem. However, unless the problem is very large (i.e., MAXIT should at least be greater than the problem dimension), MAXIT should not be set to an arbitrarily large number. Where the problem has been run and the MAXIT limit hit, the results of such a run should be examined before the limit is changed. The equation solver may, in fact, be having great difficulty with the problem, and simply increasing the limit of MAXIT may produce only an even longer unconverged computer run.

The following basic behavior patterns can indicate that the equation solver is having difficulties with the problem:

1. During the iterations, the values of the root mean square of the constraint violations do not seem to be getting smaller. The value

*One typical error that a new user might make would be to define a subsystem with a constraint (say, some flow quality) without setting ACC to some smaller value (say, 10^{-4}). Quality being in the normal range of zero to one would mean that (without ACC redefined) a solution would be found with quality constrained to within only 0.1 of its required value.

**The scaling can be changed by selecting a new starting point with a different Jacobian.

of the root mean square of the constraint violations is printed out, along with the iteration number, as "N=" and "F=" whenever PRINT>0.

2. The equation solver's parameter BASE, which is printed out when PRINT>1, is not getting smaller with the iterations. The value of BASE is the sum of the squares of the scaled constraint violations. If BASE is getting smaller but the root mean square is not, it may mean that there is still no difficulty and the maximum number of iterations could be increased to obtain convergence.
3. The equation solver's parameter MU is becoming larger. The parameter MU, which also is printed out whenever PRINT>1, represents a measure of the relative weight between the steepest-descent step and the quasi-Newtonian step. If MU is zero, a pure quasi-Newtonian step is taken. As MU increases, a smaller and smaller step is taken, weighted more and more in the direction of a steepest-descent step. The steepest-descent step is along the negative gradient of the root mean square of the constraint violations (as a function of the varied parameters). A value of MU greater than five or six is considered large. If MU is extremely large (say, ten or more), the problem that has been set up is probably near singular (that is, the constraints are almost linearly dependent). In such a case, it would be unwise to simply increase MAXIT to try to obtain convergence; the physical problem should be reconsidered.

The equation solver also may fail if it runs into one of the user-imposed bounds. These bounds are specified, along with the parameters being varied, within the VARY statement. The equation solver can only work with equality constraints; additional inequality constraints on the independent variables of the problem act only as a safeguard to prevent extremely large variations of the parameters. The equation solver will attempt to locate the root of the equations within these bounds. Of course, no such root may exist.

Where no root exists within the bounds, the bound being hit is indicated by printing out an array, Y, that represents the scaled values of the VARY parameters. These Y values are such that the lower and upper bounds are rescaled to 0 and 1. Thus, those parameters greater than their upper bounds have their corresponding Y values greater than 1, and those parameters lower than their lower bounds have Y values of less than zero. The relative positions between the bounds of all other variables are also indicated by the Y array. Thus, the bound that is causing the problem can be determined and changed within the VARY statement.

It should not be assumed, where a bound is hit, that the root lies outside of the bounds. At least for problems with dimension greater than one, it is possible that the root to the constraint equations does lie within the bounds. However, due to the technique employed by the equation solver, the iterations may tend to drive some of the

parameters out of the bounded region (and, later, back into the region) in order to find the solution. Thus, even though a bound on a particular variable might have been enlarged, the equation solver ultimately may find a root within the original bounds. If such were the case, changing the initial starting value of the iterations might have located the root without the necessity of enlarging one or more of the bounds.

Initially, the equation solver will attempt to determine the Jacobian of the constraint equations by the finite-difference method. This Jacobian is then inverted to determine the inverse Jacobian. After this initial inversion, no more matrix inversions are performed; the Jacobian and inverse Jacobian are simply updated as the iterations proceed. At least at the initial parameter values, then, the constraints should not be linearly dependent and the Jacobian should be of full rank. Two simple failure modes are easily checked at this point:

1. A particular constraint is independent of all the VARY parameters within the subsystem; this would yield a row of zeros within the Jacobian.
2. No constraint is influenced by varying one of the parameters; this would yield a column of zeros within the Jacobian.

Both of these problems usually result from the physical problem itself, in which case the problem constraints must be examined more carefully. Alternatively, the perturbations of the VARY parameters used in determining the Jacobian may have been too small to produce a change in the violations. These perturbations are evaluated using the DEL parameter specified in the SWITCH statement. The perturbations of each VARY parameter are calculated as the maximum of DEL and DEL times the difference in the upper and lower bounds on the parameter. DEL, at present, defaults to 10^{-6} . If a particular constraint's initial violation is very large and there is only a weak influence (i.e. small derivative) on the VARY parameter, the finite-difference calculation of the Jacobian element may be wrong. This error can sometimes be rectified by enlarging DEL, but this procedure can produce inaccuracies in the approximation of the Jacobian elements for those constraints that are reasonably scaled. It may be necessary to locate a better starting value, where the initial constraint violations are smaller. However, such poorly scaled problems rarely occur.

Another termination mode is possible with poorly scaled problems when the independent variables have to change by an extremely small value in order to drive the root mean square of the constraint violations below ACC. In these cases, the independent variables actually are almost at the root to the constraint equations anyway; thus, such a termination may be as good as a normal termination.

Only one other termination mode exists for the equation solver. This mode is indicated by the message "unable to find an improved point," which occurs when the technique cannot find a reduced value of the root mean square of the scaled constraint violations when even a small step is taken in the direction of the steepest descent. This error can occur when the nonlinearities of the constraints are too difficult for the equation solver. The equation solver tries to update the Jacobian and inverse Jacobian,

and after many iterations these attempts will accumulate numerical inaccuracies. Thus, the calculated steepest-descent direction may be wrong. Although it would be possible to restart at this point and form a new Jacobian, the present technique does not do this; termination of the problem is preferred, with restart initiated manually after the failure has been examined. This type of termination is the hardest type to resolve, but usually it can be resolved by decomposing the problem into several sets of smaller-dimension problems. Once a solution is obtained, the problem can be reformulated so that convergence problems are not encountered. However, good reformulations require considerable experience and judgment.

6.2.2 Optimization Tasks

Many of the failure modes for the optimizer are similar to those for the equation solver. For example, one can run into the maximum number of iterations. Again, the iterations should be reviewed before MAXIT is simply increased. The same four parameters -- ACC, DEL, MAXIT, and PRINT -- are used to control the iterations and print out from the optimizer. The optimizer also can terminate as a result of convergence of the independent variables. As with the equation solver, this outcome might be sufficiently close to the solution -- however, for optimization problems it is much more difficult to tell.

Normal termination for the optimizer is determined by the value ACC. The iterations terminate when the following condition is satisfied:

$$|\text{GRAD}(F)*D| + \text{SUM}(|LM*C|) < \text{ACC}$$

where GRAD(F) is the gradient of the objective function, D is the latest vector change in the independent variables, LM is the vector of Lagrangian-multiplier estimates, C represents the constraint violations, and SUM() represents summation. The constraint violations are defined (as with the equation solver) even for "greater-than" inequality constraints. For "less-than" inequality constraints, however, the right- and lefthand expressions of the constraint are reversed in evaluating the constraint violation. The value on the lefthand side of the above inequality is denoted "L" and is printed out along with the iteration number "N," the objective function value "F," the independent variable values "X," and the constraints "C," whenever PRINT is set greater than zero.

The termination condition specified above is often used. Because this condition does not depend on estimates of LM (which are calculated by the code), however, it may be difficult at times to define a reasonable value of ACC for termination. It may, in fact, be necessary to run the problem once with some small value (say, 10^{-3}) for ACC and a small value (say, five times the problem dimension) for MAXIT, and then determine whether or not ACC should be changed. For example, after running a problem, if the printed values of "L" are very large (say, 10^{10}), reasonable convergence may be obtained for an ACC value of 10^4 . Alternatively, if the initial "L" were equal to 10^{-2} , ACC might have to be 10^{-6} for reasonable convergence.

The optimizer can fail if it runs into a location where the linear approximations to the constraints are locally dependent. This type of failure, detected within the

quadratic programming procedure called by the optimizer, is rather rare because of the way the constraints are handled internally. The only solution to this termination would be to restart the problem at another point. (The physical problem should also be reviewed to detect any abnormality.)

Unlike the equation solver, the optimizer will not stop if a variable is running into a bound, because the optimizer has more degrees of freedom to work with in attempting to establish the constraints. However, it is possible to define inequality constraints for which no feasible region (i.e., the region in which all inequality constraints are satisfied) exists. The optimizer will try to find the feasible region; if it cannot do so, it will terminate with the message "unable to find a feasible point." Problems with only equality constraints may also run into this problem, because all independent variables have bounds, which are additional inequality constraints. In this case, the equality constraints cannot all be solved simultaneously within this bounded region.

Two other failure modes exist for the optimizer. Both modes involve efforts to find an improved point, starting along a search direction from the current point. This search direction, determined by the quadratic subprogram, depends on the built-up Hessian of the Lagrangian of the problem and on the finite-difference representations of the gradients of the objective function and the constraints. The finite-difference representations of the gradients are obtained (as in the equation solver) using the perturbations of the independent variables by the parameter DEL. Thus, DEL needs to be small enough for an accurate representation of the gradients and large enough to avoid round-off errors. However, the gradients of the objective functions and constraints are calculated more than once during the optimizer's iterations. The value of DEL used will affect all the iterations, which is unlike the situation with the equation solver (where DEL is used only for the initial Jacobian calculation).

The first failure mode associated with the line search involves having no "downhill" direction in which to go. This difficulty sometimes occurs in later iterations, when the accumulated numerical errors prevent the calculation of a good search direction. Sometimes adjusting DEL to a smaller number and starting over can correct this problem. The other mode involves an inability to locate an improved point along the line (as measured by a line-search function). This difficulty also sometimes occurs during later iterations and may be resolved by reducing DEL. Often, when this failure mode is encountered, one is approaching a solution. The optimizer will attempt to find an improved point along the search direction by pulling back towards the current point five times; if it fails after five attempts, a message is printed and the optimization terminates.

As was the case with the equation solver, when the PRINT switch is set to 2, 3, or 4, additional information concerning the numerical iterations is printed out. When PRINT is set to 2, information about the values of the Lagrangian multipliers, the displacements to be attempted in the independent variables, and the initial and final infeasibility norms is also printed. Higher values for PRINT will cause details of the iterations within the quadratic programming routine to be printed out; these details are not discussed here.

6.3 FAILURES DUE TO PHYSICAL PROBLEMS

Most of the failures due to numerical procedures discussed above would not occur if reasonable problems were set up. In fact, most of the difficulties users have in running the SALT code are caused by the actual problems that they define. These difficulties can take many forms. Those forms that new users of the SALT code tend to encounter are discussed here.

In general, when a problem cannot be made to converge after many attempts, it is best to remove all the constraints and then restore them one at a time, with a separate computer run for each constraint added. Using a parameter sweep rather than a VARY statement may also help to determine the parameters' effects on the system; this information can be of use in reformulating the problem.

Some of the less severe physical problems that arise will often cause the equation solver or optimizer to print a failure message that immediately identifies the problem. For example, it is often desired to constrain the heat-exchanger exit temperature to some specified value by varying the heat load on the exchanger. This constraint will work, provided that the fluid within the heat exchanger is in a single phase. However, if the fluid enters the two-phase region, then the exit temperature will be a function of the exit pressure, and many different values of the heat load will result in the same temperature (probably not the one desired). In this case, if the heat load initially caused the fluid to be in the two-phase region and this constraint was the only one within the subsystem, the equation solver would fail initially with the message that the constraint was independent of the varied parameter. On the other hand, had the initial value of the heat load caused the fluid to be in a single-phase region, initially varying the heat would yield some functional dependence on the exit temperature, but the equation solver might later generate heat values for which the exit temperature displayed no functional dependence. In this case, the equation solver might fail.

Problems in which a fluid enters a two-phase region are best handled by constraining those thermodynamic properties that are still independent within the region. Thus, constraints on the exit enthalpy or quality from the heat exchanger would be well posed even in the two-phase region. These multiphase fluid constraints may take other forms. Rather than varying the heat load on the heat exchanger, the mass flow rate for a fixed heater load might have been varied; the same problem would occur. If additional parameters were being varied within the same subsystem to establish additional constraints, the equation solver might persist in its efforts for a considerable time before failing, and it might be difficult to determine by looking at the iterations exactly why it failed.

A problem similar to the two-phase problem is that functional dependence of a constraint on a parameter can be lost. This can occur when a model parameter is bounded. For example, any variable that is cut off if it lies above or below some fixed bound may cause constraints dependent on that variable to be independent of the parameters being varied. This problem can result whenever the user defines constraints using MAX or MIN functions, or when such functions are used within models. Had the log mean temperature difference calculated within the heat-exchanger model been defined to be zero whenever the hot- and cold-temperature profiles crossed over, it would be

extremely difficult to constrain that variable to a specific value by varying some other parameter within the system. Another example involves constraining steam quality at some point within the system. Usually, steam quality is defined to be between 0 and 1; if, during the iterative process, the steam is superheated, then the quality is 1, and the constraint will not show any functional dependence on the parameter being varied. The solution to this problem (at least for subcritical pressures) is simply to let steam quality be defined as a continuous quantity even above and below the two-phase region. Such a definition is used in the SALT code.

The loss of functional dependence of a constraint on a variable being varied can also occur because of the arrangement of the system models. For example, in a liquid-metal MHD system it might be necessary to have a recycle-liquid loop. In order to close the pressure at the recycle point, one might vary the exit pressure out of some model in the liquid's flow path, only to set the exit pressure out of some other model further along the path later. In this case, matching the pressure at the end of the path to that at the beginning would be impossible; the varied exit pressure would have no effect on the pressure at the end of the path. The equation solver should clearly indicate that the pressure constraint is independent of all varied parameters within the subsystem. It is assumed that no other varied parameter affects the pressure downstream of the model having the set exit pressure. In this case, there may be no message from the equation solver, and one may have set up a problem with N constraints but with only $N-1$ variables that really have any sufficiently strong influence on the constraints. Varying the pressure might weakly affect one of the other constraints, so that there would be no immediate termination message. This type of problem can be extremely difficult to track down, even using the iteration output generated by the equation solver. One thing to look for in problems such as this one is a very great increase in the value of MU , because such problems will generate an almost singular Jacobian.

Another situation of this type can result from the mixing of flows. The simplest flow mixer may take the exit pressure as the minimum of the input flow pressures. For such a model, varying some parameter affecting the pressure upstream of one of the mixer's input flows may or may not affect the mixer's outlet pressure. If the initial value of such a parameter influenced the mixer's exit pressure, but during later iterations it did not (due to the other input flow having a lower pressure), any constraint dependent on the mixer's exit pressure would be difficult to converge on. The solution to this problem is, of course, to have the flow-mixer model actually reflect both input flow pressures when calculating the output flow. Even so, the larger input pressure may have only a weak influence on the exit pressure.

Essentially, the examples considered so far have involved the loss of functional dependence of the constraints on the varied parameters. The situation can also occur where the user has imposed several constraints to be established by variation of several parameters and the constraints are linearly dependent. In this case, more parameters exist than are needed, and one of the constraints and parameters should be removed. The problem would have an infinite number of solutions. This sometimes occurs when one has inadvertently tried to constrain both quality and enthalpy at the same point of a flow stream. Problems such as these will definitely make the equation-solver parameter MU grow large, because such problems are singular. Sometimes, by examining the variables

and constraints that change the most as MU grows large, one can isolate the constraints that are dependent.

Inexperienced users of the SALT code sometimes generate several other common problems that may not hinder convergence at all. These problems generally arise with use of the multiple-entry models. For instance, the heat-exchanger model should never have one entry called for a given value of the heat load and then have the other entry called later within a subsystem in which the heat load is different; the influence of the changed heat load on the first-entry flow would never be accounted for.

Another instance occurs when the heat exchanger is called with the parameter T_SET set for the incorrect first entry. If T_SET(2) were specified and the hot entry called first, an incorrect heat load would be used. Similar mistakes may be made with other multiple-entry components. Multiple-entry models have their advantages, but they should be used with caution.

The decomposition of system configurations into nested subsystems can cause convergence problems at times. For example, if the values of parameters that satisfy constraints in an inner subsystem also affect the constraints in the outer subsystem, then whenever the inner subsystem fails to converge, problems can be expected in the outer system. Convergence within the inner subsystem should be kept at least as tight as in the outer system; otherwise, the calculation of gradients by the finite-difference method may be incorrect. It is possible with nested systems for the final iteration of all subsystems to converge even if some of the inner systems fail to converge for some of the iterations of the outer systems.

Most of the failure modes discussed here have been encountered by analysts not totally familiar with systems-analysis concepts. The use of SALT has been valuable in helping to increase understanding of system performance, and very few failures have occurred as results of "bugs" in the SALT code. However, very few codes are truly "bug-free." The authors would appreciate being informed of any problems encountered by users of the SALT code.

7 ADDING NEW MODELS AND FLOW TYPES

7.1 THE INTERFACE FILE

The SALT code has been designed so that new models can be added as effortlessly as possible, while models with arbitrary levels of complexity are still handled. The key to maintaining the flexibility of handling arbitrary models lies in the mechanism used for interfacing the constructed driver with the component models. Numerous approaches have been employed in this interfacing problem; the one ultimately used in the SALT code is relatively simple and has proven to be one of the most flexible. The technique is simply to read in from an external file those variables that must be declared for each component model used in a given system analysis. These variables may be the names of model parameters that are passed to the component models, the names of entry variables, or the names of other variables that simply need to be included whenever a particular model is used.

The file that contains this interfacing information, called "INTF," is used by the SALT code when constructing the PL/I driver for the problem under consideration. The form of the interface consists of a series of header statements -- each with a 0, 1, 2, or 3 in the first column -- followed by other data that depend on the number. The type-0 statements are used to read in information that will be needed by the system models in locating specific substructures of the models, the type-1 statements define the model interfaces, the type-2 statements define the flow interfaces, and the type-3 statements define additional coding that will be unconditionally inserted into the PL/I driver.

The specific form of the type-0 statement is as follows:

```
OSUBSTRUCTURE_NAME  SUBSTRUCTURE_HEAD_PTR
```

Here, SUBSTRUCTURE_NAME is the name of the model substructure that will be included within a linked list for use in system models, and SUBSTRUCTURE_HEAD_PTR is a pointer variable that points to the beginning of this linked list.

Some of the substructures to be included in a linked list have variable names -- such as FLC, FLH, FLC1, etc. -- so the character "*" may be used at the end of a SUBSTRUCTURE_NAME to refer to any substructures that begin with the specified SUBSTRUCTURE_NAME. All such substructures starting with those common characters will then be included in the same linked list. Thus, to create a linked list for all of the flow substructures starting with the characters "FL," one would write the following:

```
OFL*    FLOW_HEAD_PTR
```

The type-1 statement header takes the form of a 1 in the first column, followed by the name of the model. Statements following this header statement represent the PL/I declarations of the model structure variable, followed by the declaration of the model entry points, each on a separate line. The interface statements for the ST model, for example, are as follows:

1ST

DCL

```

1 ST BASED,
  2 NAME CHAR(16),
  2 FLC1,
    3 FNAME CHAR(16),
    3 ID CHAR(4),
    3 ATOM(8) FLOAT(16),
    3 PROP,
      4 (TEMP,PRES,ENTH,ENTP,QUAL,RHO,VEL,MASS)    FLOAT(16),
    3 COMP,
      4 (XAR,XCH4,XCO,XCO2,XH,XH2,XH2O,XH2S,XK,XKOH,XNO,XN2,
        XO,XOH,XO2,XSO2,XHCL,XCH3OH,XC,XCOS,XNH3,XS,XCL) FLOAT(16),
    3 SOL,
      4 WTF FLOAT(16),
2 FLC2,
  3 FNAME CHAR(16),
  3 ID CHAR(4),
  3 ATOM(8) FLOAT(16),
  3 PROP,
    4 (TEMP,PRES,ENTH,ENTP,QUAL,RHO,VEL,MASS)    FLOAT(16),
  3 COMP,
    4 (XAR,XCH4,XCO,XCO2,XH,XH2,XH2O,XH2S,XK,XKOH,XNO,XN2,
      XO,XOH,XO2,XSO2,XHCL,XCH3OH,XC,XCOS,XNH3,XS,XCL) FLOAT(16),
  3 SOL,
    4 WTF FLOAT(16),
2 PARM,
  3 DDNAME CHAR(7),
  3 MODE CHAR(15),
  3 EXIT_PRES FLOAT(16),
  3 EFFICIENCY FLOAT(16),
  3 MECH_EFF FLOAT(16),
  3 SR FLOAT(16),
  3 EXT_MASS FLOAT(16),
  3 FLOW_FACT FLOAT(16),
  3 EXHAUST_LOSS FLOAT(16),
  3 DM FLOAT(16),
  3 WV FLOAT(16),
  3 WHEEL_SPEED FLOAT(16),
  3 CONS FLOAT(16),
  3 VOL_FLOW_RATE FLOAT(16),
  3 PRINT_FIXED_BIN(15),
2 POWER,
  3 (INPUT,PRODUCED,CONSUMED,LOSS) FLOAT(16),
2 E_LOSS,
  3 PTR POINTER,
2 COST FLOAT(16);
DCL STC ENTRY;
DCL STOUT ENTRY;

```

Other variables may be declared within the interface for each model, but no particular variable should be declared in more than one model interface. If other variables are declared, they should follow the declaration of the model structure variable.

Although the statements following the header are simply PL/I declarations, some checking of this input is done by SALT in order to retain the names of the 2-level substructures and the entry names. In the case of the model declarations, some programmers write the comma before the next line rather than after the line. The SALT code checks for the presence of the string " 2 " in locating these 2-level substructures, but it will not properly locate those preceded by ",2."

Usually, no executable statements appear within the interface; however, it is possible to include whole PL/I procedures. Such procedures are compiled right into the PL/I driver code, along with the other statements within the interface. Only those statements belonging to the interfaces of models actually used in the system problem will be brought into the driver code.

The type-2 statement header takes the form of a 2 in the first column, followed by the name of a flow type. Statements following this header declare the form of the flow. For example, the interface statements for the STM flow are as follows:

```
2STM
DCL 1 STM BASED,
    2 NAME CHAR(16),
    2 ID CHAR(4),
    2 ATOM(8) FLOAT(16),
    2 PROP,
      3 (TEMP,PRES,ENTH,ENTP,QUAL,RHO,VEL,MASS)    FLOAT(16),
    2 COMP,
      3 (XAR,XCH4,XCO,XCO2,XH,XH2,XH2O,XH2S,XK,XKOH,XNO,XN2,
        XO,XOH,XO2,XSO2,XHCL,XCH3OH,XC,XCOS,XNH3,XS,XCL) FLOAT(16),
    2 SOL,
      3 WTF FLOAT(16);
```

Like the model interfaces, only those flow-type interfaces actually used in the system problem will be included in the driver code.

The type-3 statement header takes the form of a 3 in the first column, followed by any comment or blanks. The lines following this header statement, up to the occurrence of another header line, will be included in the driver code. For example, declarations of unit-conversion variables might be included.

The different types of interfaces may be freely mixed. For readability, however, it is useful to group all of the different types together. The INTF file may also reside in more than one data set and be concatenated together at run time.

7.2 ADDING NEW MODELS

New PL/I component models may be added to the library of SALT models at any time. One has only to develop the new model, taking into account a few rules necessary for the SALT code to interface correctly with it, and to add the appropriate interface statements to the INTF file.

Three basic rules are used in developing a new model. First, a model may have any number of entry points, but the first several characters of the entry name should be the name of the model as it appears within a PROCESS statement. The additional letters, up to the PL/I limit of seven characters for entry names, define the separate entry points. For example, the heat-exchanger model (HX) has the entry points HXH, HXC, and HXOUT. When a model is referred to within a PROCESS statement without the use of the colon and entry name, the "C" entry (as in HXC) will be called. The main calculational entry should be specified as this "C" entry when new models are being developed.

The second basic rule concerns the arguments that are passed to these entry points. The first argument to all entry points is a pointer to the model structure variable. The additional arguments that follow are pointers to the flow variables, which should be arranged in the order of pass-through flows, input flows, and output flows to the model. For the "OUT" entry (e.g., STOUT), the only argument should be the model structure variable.

The third rule concerns the model structure variables themselves. The 1-level name should be the name of the model. The first 2-level name should be NAME, declared as a CHAR(16) variable. This variable is always defined by the SALT code as the name of the model (including user-defined label) as it is called within the system problem.

Any other 2-level structures may be defined by the model developer to store various input and output data from the model. The 2-level name POWER already has a special structure; if used, it should correspond to that used by the existing models.

Beyond these three basic rules concerning the naming of the entry points, the arrangement of the arguments, and the naming of the model substructure variables, any type of PL/I coding may be used within the model. A model may even call FORTRAN subroutines to perform its calculations. However, the output from a model must be a function of only the input flows and model parameters. Each time a model is called with the same input values, it should return the same output values.

Once a model has been developed and debugged, it need only be compiled into the model load library and the interface statements added to the INTF file. Usually, the interface file will consist of the model structure variable and the declaration of the entry points. The model structure has already been written, so it will need only to be copied from the model into the interface file, with the possible addition of initial attributes to define default input values. These additions to the INTF file should take only a few minutes of editing time.

7.3 ADDING NEW FLOW TYPES

New flow types to be processed by newly developed models may be added to the SALT code at any time ("flow type" refers to the structure of the flow variables). Additional steam, liquid, or gas flows that are structurally the same as STM, LIQ, or GAS are generated as needed within a system problem using the labeling option for flows (i.e., STM_1, STM_2, etc.). The existing flows of STM, GAS, and LIQ are technically of the

same type, but they have been defined as separate flow types to furnish the user of SALT with some variety in the flow names to be used.

The addition of a new flow type is accomplished by adding the PL/I declaration of the flow variables to the INTF file. Of course, this new flow will not be usable unless new component models have been written to accommodate the new flow type. If system models exist that print out flows, then these models may have to be modified to accept the new flow types. Nothing further needs to be done to add a new flow type.

Each flow variable, like the model structures, should have as its first 2-level element the variable NAME, declared as a CHAR(16) variable. This variable will be assigned the name of the flow as used within the system problem by the SALT code.

REFERENCES

1. Cook, J.M., *User's Guide for GSMP, A General Systems Modeling Program*, Argonne National Laboratory Report ANL/MHD-79-11 (1979).
2. Berry, G.F., and J.M. Cook, *Application of a General System Modeling Program*, *Advances in Engineering Software* 5(4):221 (1983).
3. Berry, G.F., J.M. Cook, and C.B. Dennis, *Application of Polyalgorithm Optimization to MHD Power Plant Design*, *Energy Systems* 2(3) (1980).
4. Geyer, H.K., *GPSAP/V2 with Applications to Open Cycle MHD Systems*, Argonne National Laboratory Report ANL/MHD-80-15 (1980).
5. Geyer, H.K., and G.F. Berry, *A Preprocessor for Performing Lumped Component Analysis*, *Energy Systems* 2(3) (1982).
6. Berry, G.F., and H.K. Geyer, *SALT - A Steady State and Dynamic Systems Code*, Proc. Third American Society of Mechanical Engineers International Computing in Engineering Conf. (1983).
7. Berry, G.F., and H.K. Geyer, *The SALT Steady-State Systems Code*, Proc. Systems Simulation Symp. on Fossil Fuel Conversion Processes, sponsored by Morgantown Energy Technology Center, Morgantown, W.Va. (Dec. 1983).
8. Berry, G.F., and H.K. Geyer, *The SALTD Dynamic Systems Code*, Proc. Systems Simulation Symp. on Fossil Fuel Conversion Processes, sponsored by Morgantown Energy Technology Center, Morgantown, W.Va. (Dec. 1983).
9. Powell, M.J.D., *A Hybrid Method for Nonlinear Equations*, in *Numerical Methods for Nonlinear Algebraic Equations*, Gordon and Breach Science Publishers, New York (1970).
10. Powell, M.J.D., *A Test Algorithm for Nonlinearly Constrained Calculations*, presented at the 1977 Dundee Conf. on Numerical Analysis, Dundee, U.K. (1977).
11. Hindmarsh, A.C., *Numerical Integration of an Initial Value Problem for a System of Ordinary Differential Equations*, Argonne National Laboratory, Applied Mathematics Division subroutine documentation (1980).

APPENDIX A: JOB-CONTROL LANGUAGE FOR IBM SYSTEM AT ANL

```

//JOBNAME = THE JOB NAME
//SYSNAME = THE SYSTEM NAME
//CLASS = THE CLASS OF THE JOB
//PRIORITY = THE PRIORITY OF THE JOB
//TIME = THE TIME OF THE JOB
//CPU = THE CPU OF THE JOB
//MEM = THE MEMORY OF THE JOB
//DISK = THE DISK OF THE JOB
//TAPES = THE TAPES OF THE JOB
//PRINT = THE PRINT OF THE JOB
//PUNCH = THE PUNCH OF THE JOB
//READ = THE READ OF THE JOB
//WRITE = THE WRITE OF THE JOB
//EXEC = THE EXEC OF THE JOB
//START = THE START OF THE JOB
//END = THE END OF THE JOB
//JOB = THE JOB OF THE JOB
//JOBNAME = THE JOB NAME
//SYSNAME = THE SYSTEM NAME
//CLASS = THE CLASS OF THE JOB
//PRIORITY = THE PRIORITY OF THE JOB
//TIME = THE TIME OF THE JOB
//CPU = THE CPU OF THE JOB
//MEM = THE MEMORY OF THE JOB
//DISK = THE DISK OF THE JOB
//TAPES = THE TAPES OF THE JOB
//PRINT = THE PRINT OF THE JOB
//PUNCH = THE PUNCH OF THE JOB
//READ = THE READ OF THE JOB
//WRITE = THE WRITE OF THE JOB
//EXEC = THE EXEC OF THE JOB
//START = THE START OF THE JOB
//END = THE END OF THE JOB
//JOB = THE JOB OF THE JOB

```

UNIVERSITY OF CALIFORNIA
LIBRARY

APPENDIX A: JOB-CONTROL LANGUAGE FOR IBM SYSTEM AT ANL

The preceding chapters have dealt with the data that must appear within the STRUCT file. This file is usually the only file that must be changed when running a new systems-analysis problem. However, other files are used by the SALT code in the process of compiling the PL/I driver that represents the system under consideration. Essentially, these other files are temporary work files or output files and are not usually saved from job to job.

Three major steps are required in running the SALT system code after the STRUCT file has been prepared. The first step is to run the SALT code itself and translate the STRUCT file into a PL/I code; the second step is to compile this code, and the third step is to actually execute the PL/I code. The performance of these three steps has been conveniently arranged in an instream job-control-language (JCL) procedure called SYSTEM for use on the ANL computer. The JCL using this procedure is as follows:

```
//JOBNAME JOB TIME=2,REGION=350K,CLASS=W,MSGCLASS=W
//*MAIN ORG=LOCAL,SYSTEM=(S33A,S33B),LINES=5
//SYSTEM PROC
//ONE EXEC PGM=SALT
//STEPLIB DD DSN=Bxxxxx.SALT.LOAD,DISP=SHR
//STRUCT DD DDNAME=STRUCIN
//INTF DD DSN=Bxxxxx.SALT.INTF,DISP=SHR
//SYSDRV DD UNIT=SASCR,SPACE=(TRK,(2,1)),DISP=(NEW,PASS)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1511)
//PLO EXEC PGM=IELOAA,PARM='NS,NA,NX,NAG,NOESD,NSTG,NOF,NOP'
//STEPLIB DD DSN=PLI.OPT.LINKLIB,DISP=SHR
//SYSIN DD DSN=*.ONE.SYSDRV,DISP=(OLD,DELETE)
//SYSLIN DD UNIT=SASCR,SPACE=(CYL,6),
// DISP=(NEW,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1511)
//SYSPUNCH DD DUMMY
//SYSUT1 DD SPACE=(CYL,6),UNIT=(SASCR)
//TWO EXEC PGM=LOADER,REGION=150K,COND=(9,LT,PLO)
//SYSLIB DD DSN=SYS1.PLIBASE,DISP=SHR
// DD DSN=Bxxxxx.SALT.LOAD,DISP=SHR
//SYSLIN DD DSN=*.PLO.SYSLIN,DISP=(OLD,DELETE)
//SYSLOUT DD SYSOUT=*,DCB=(RECFM=FB,LRECL=121,BLKSIZE=1573)
//SYSPNCH DD DUMMY
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1511)
//SYSPUNCH DD DUMMY
// PEND
// EXEC SYSTEM
//ONE.STRUCIN DD *
```

(contents of file STRUCT)

These JCL lines, which carry out the three basic steps referred to above, are briefly described here.

The first line of any JCL, the JOB card, specifies the maximum time (in minutes) that a job is permitted to run on the computer and the amount of main core used. The parameter CLASS defines the priority of the job and may take the values of U (for highest priority), W (for normal priority), X (for overnight service) or Y (for weekend service). The MSGCLASS parameter may be set to W to fetch the output at the computer terminal or to A to print the output on a line printer.

The second line specifies on-line printer destinations, the computer used, and the maximum number of output lines. If a large number of parameter sweeps are to be performed, the LINES parameter (which specifies the maximum number of lines in thousands) may need to be increased.

The next line specifies the beginning of the SYSTEM procedure. The following group of six lines carries out the first step, the translation of the STRUCT file. Here, STEPLIB is the data set containing the SALT code, INTF is the interface file, SYSDRV is the output file containing the generated PLI code, and SYSPRINT contains a reflection of the STRUCT file and possible error messages.

The next eight lines, starting with //PLO, accomplish the compilation of the PLI code. In this case, STEPLIB refers to the data set containing the PLI compiler, SYSIN is the PLI code generated in the first step, SYSLIN is the compiled code, SYSPRINT contains error messages from the compilation, SYSPUNCH is not used, and SYSUT1 is a work file used by the compiler.

The next group of eight lines carries out the final step required in running the compiled code. The SYSLIB file contains the concatenation of several data sets representing the components of the system and various IBM-supplied procedures, such as SIN, COS, ABS, etc. (These are in SYS1.PLIBASE for PLI codes.) The component models and other mathematical procedures are referenced by the next two lines. Here, SYSLIN refers to the compiled PLI code, SYSLOUT contains the loader map and loader error messages, SYSPNCH and SYSPUNCH are not used, and SYSPRINT contains the major output for the system analysis.

Finally, the next two lines close the instream procedure (//PEND) and execute this procedure (//EXEC SYSTEM). For most systems-analysis problems, the above JCL need not be changed from job to job. The rest of the JCL represents the STRUCT file, preceded by the //ONE.STRUCIN DD * line.

APPENDIX B: ABBREVIATIONS FOR KEY WORDS

The following table lists the abbreviations for the key words used in the text. The abbreviations are listed in the left column, and the full names are listed in the right column.

The table is organized alphabetically by the first letter of the abbreviation.

Abbreviation

Full Name

Abbreviation

APPENDIX B: ABBREVIATIONS FOR KEY WORDS

Abbreviation

Full Name

APPENDIX B: ABBREVIATIONS FOR KEY WORDS

The SALT code will accept abbreviations for some of the key words. These key words and their abbreviations are as follows:

Key Word	Abbreviation
PROCESS	PROC
SYSBEG	SYSB
SYSEND	SYSE
CONSTRAIN	CONS
CONTROL	CONT
MINIMIZE	MINI

APPENDIX B: ALPHABETICALLY BY KEY WORD

The table lists the key words associated with some of the key words. These key words and their associations are as follows:

Key Word	Association
ALPHA	ALPHA
BETA	BETA
GAMMA	GAMMA
DELTA	DELTA
EPSILON	EPSILON
ZETA	ZETA
ETA	ETA
THETA	THETA
IOTA	IOTA
KAPPA	KAPPA
LAMDA	LAMDA
MU	MU
NU	NU
Xi	Xi
OMICRON	OMICRON
PICHA	PICHA
RHO	RHO
SMALL	SMALL
TAU	TAU
Upsilon	Upsilon
PHI	PHI
CHI	CHI
PSI	PSI
OMEGA	OMEGA

**APPENDIX C: UNITS USED
IN SALT MODELS**

APPENDIX C: UNITS USED IN SALT MODELS

The units used throughout in the SALT models are SI units, with the exception of pressure (specified in atmospheres). Those parameters (such as efficiency) commonly expressed in percent are specified as fractions. Thus, 88% would be input as 0.88. Angles (such as gravitational angles) are in degrees.

APPENDIX D: FOSSIL/STEAM POWER PLANT

PROCESS GP_1:IN

SYSSEG A

PROCESS

```

LIQ_1-> IN_H2O SD_1->STM_1
STM_1-> HX_SH:C ST_HP1->STM_HP1
          ST_HP2->STM_HP2
          HX_FH:C ST_IP->STM_IP
          ST_LP1->STM_LP1
          ST_LP2->STM_LP2 ST_LP3->STM_DUM
          SC_1
STM_HP1-> FH_HP1:H <-STM_DUM
STM_HP2-> FH_HP2:H <-STM_HP1
STM_LP1-> FH_LP1:H <-STM_HP2
STM_LP2-> FH_LP2:H <-STM_LP1
STM_1-> HX_SC <-STM_LP2
          PUMP_SC HX_ECON:C FH_LP2:C FH_LP1:C
          DEAR_1 <-STM_IP
          PUMP_FH FH_HP2:C FH_HP1:C
          HX_FH <-STM_1
LIQ_1-> PUMP_BFP HX_BOIL:C IN_H2O:CYCL

```

```

VARY HX_BOIL.HEAT = * 1E1 25E6
CONS IN_H2O.DH = 0.0
VARY PUMP_BFP.EXIT_PRES = * 100 200
CONS IN_H2O.DP = 0.0
VARY ST_IP.SR = * 0.01 0.20
CONS DEAR_1.PARM.QUAL = 0.0

```

SYSEND A

PROCESS

```

AIR_1-> IN_AIR CP_AIR HX_AIR:C
FUEL_1-> INF_COAL DRY_1 CB_1 <-AIR_1->GAS_1
GAS_1-> HX_BOIL:H HX_SH:H HX_RH:H
          HX_AIR:H HX_ECON:H DRY_1:H SK_1
NULL-> SYST_1 *_*OUT

```

DATA

```

IN_AIR.PARM .ID='GAS'; .T=298.15; .P=1.0; .M=14.0;
            .XN2=0.78; .XO2=0.22;
IN_H2O.PARM .ID='H2O'; .T=0.0; .P=180.0; .M=75.0;
            .Q=0.20;
INF_COAL.PARM .MASS=2.0; .C=0.5213; .H=0.060; .O=0.3152;
            .N=0.0079; .S=0.0085; .H2O=0.227;
            .ASH=0.0371; .HHV=20.743E6;
SD_1.PARM .QUAL=0.20;
CP_AIR.PARM .EXIT_PRES=1.15; .EFFICIENCY=0.83;
CB_1.PARM .ASH_DET=0.0;
DRY_1.PARM .H2O_DET=0.05;
HX_BOIL.PARM .HEAT=12E6;
HX_SH.PARM .T_SET(2)=811.;
HX_RH.PARM .T_SET(2)=811.;
HX_AIR.PARM .T_SET(2)=500.;
HX_ECON.PARM .HEAT=1E5;
ST_HP1.PARM .EXIT_PRES=100.; .EFFICIENCY=0.84; .SR=0.10;
ST_HP2.PARM .EXIT_PRES=50.; .EFFICIENCY=0.84; .SR=0.10;
ST_IP.PARM .EXIT_PRES=15.; .EFFICIENCY=0.86; .SR=0.07;
ST_LP1.PARM .EXIT_PRES=5.; .EFFICIENCY=0.87; .SR=0.05;
ST_LP2.PARM .EXIT_PRES=1.; .EFFICIENCY=0.87; .SR=0.05;
ST_LP3.PARM .EXIT_PRES=0.066; .EFFICIENCY=0.87; .SR=0.0;
SC_1.PARM .EXIT_PRES=0.066;
PUMP_SC.PARM .EXIT_PRES=15.0; .EFFICIENCY=0.90;
PUMP_FH.PARM .EXIT_PRES=180.0; .EFFICIENCY=0.90;
PUMP_BFP.PARM .EXIT_PRES=190.0; .EFFICIENCY=0.90;
SYST_1.PARM .POWER_HEAD_PTR=POWER_HEAD_PTR;
            .FLOW_HEAD_PTR=FLOW_HEAD_PTR;

```

LOOP: A N= 1 F= 1.2460E+10
 X= 1.2000E+07 1.9000E+02 7.0000E-02
 C= -1.1162E+05 8.1000E+00 -6.3133E-02

LOOP: A
 S= 4.5277E+05 9.9000E-01 1.1525E+00
 MU= 0.00000E+00

LOOP: A N= 5 F= 3.4243E+02
 X= 1.8625E+07 1.8182E+02 1.2478E-01
 C= -1.8505E+01 1.8750E-10 6.4744E-10
 SCALE TERMINATION, ACTUAL= 1.67040E-09 IN LOOP: A

IN_H2O

ID=H2O
 TEMP = 6.31148E+02
 PRES = 1.80000E+02
 VEL = 0.00000E+00
 ENTH = 1.89132E+06
 MASS = 7.50000E+01

SD_1

QUALITY = 2.00000E-01

HX_SH

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 15.45 15.00 KG/S
 INLET TEMPERATURES = 1665.08 631.15 K
 AVERAGE TEMPERATURES = 1343.92 721.07 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LCG MEAN TEMP DIFFERENCE = 5.93089E+02 K
 HEAT TRANSFERRED = 1.32690E+07 W
 HEAT TRANSFER SURFACE AREA = 2.23728E+04 SQ-M
 HEAT FLUX = 5.93089E+02 W/SQ-M
 SURFACE TEMPERATURES = 1071.99 1071.99 K

ST_HP1

MODE = DESIGN
 TURBINE EFFICIENCY = 8.40000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 2.18482E+06
 FLOW FACTOR = 2.36530E-05
 DESIGN MASS FLOW RATE = 1.50000E+01
 SPLIT RATIO = 1.00000E-01
 VOL FLOW RATE = 4.36901E-01
 EXHAUST LOSS = 0.00000E+00

ST_HP2

MODE = DESIGN
 TURBINE EFFICIENCY = 8.40000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 2.09772E+06
 FLOW FACTOR = 3.57563E-05
 DESIGN MASS FLOW RATE = 1.35000E+01
 SPLIT RATIO = 1.00000E-01
 VOL FLOW RATE = 6.93773E-01
 EXHAUST LOSS = 0.00000E+00

HX_RH

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 15.45 12.15 KG/S
 INLET TEMPERATURES = 1022.75 624.86 K
 AVERAGE TEMPERATURES = 878.57 717.93 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 1.55072E+02 K
 HEAT TRANSFERRED = 5.47389E+06 W
 HEAT TRANSFER SURFACE AREA = 3.52990E+04 SQ-M
 HEAT FLUX = 1.55072E+02 W/SQ-M
 SURFACE TEMPERATURES = 867.68 867.68 K

ST_IP

MODE = DESIGN
 TURBINE EFFICIENCY = 8.60000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 3.87820E+06
 FLOW FACTOR = 6.89866E-05
 DESIGN MASS FLOW RATE = 1.21500E+01
 SPLIT RATIO = 1.24777E-01
 VOL FLOW RATE = 2.32286E+00
 EXHAUST LOSS = 0.00000E+00

ST_LP1

MODE = DESIGN
 TURBINE EFFICIENCY = 8.70000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 2.54706E+06
 FLOW FACTOR = 1.77638E-04
 DESIGN MASS FLOW RATE = 1.06340E+01
 SPLIT RATIO = 5.00000E-02
 VOL FLOW RATE = 4.91983E+00
 EXHAUST LOSS = 0.00000E+00

ST_LP2

MODE = DESIGN
 TURBINE EFFICIENCY = 8.70000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 2.69830E+06
 FLOW FACTOR = 4.53640E-04
 DESIGN MASS FLOW RATE = 1.01023E+01
 SPLIT RATIO = 5.00000E-02
 VOL FLOW RATE = 9.15648E+00
 EXHAUST LOSS = 0.00000E+00

ST_LP3

MODE = DESIGN
 TURBINE EFFICIENCY = 8.70000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 3.22887E+06
 FLOW FACTOR = 1.82963E-03
 DESIGN MASS FLOW RATE = 9.59715E+00
 SPLIT RATIO = 0.00000E+00
 VOL FLOW RATE = 9.65782E-02
 EXHAUST LOSS = 0.00000E+00

SC_1

EXIT PRESSURE = 6.60000E-02

FH_HP1

HEAT= 2.78348E+06
 SUBCOOL= 5.55000E+00
 AREA= 1.94188E+01
 TTD= 2.76267E+01
 DCTD= 6.05282E+01
 HDP= 1.00000E-02
 CDP = 3.00000E-03 3.00000E-03 3.00000E-03
 AREAS= 1.43913E+01 5.02755E+00 4.18492E-01
 HEATS= 6.08551E+05 2.12486E+06 5.00694E+04
 US = 4.32102E+02 7.17251E+03 1.90172E+03
 LHTDS= 9.72430E+01 5.89255E+01 6.29126E+01
 HTEMP= 7.15632E+02 6.01506E+02 5.84362E+02 5.78812E+02
 CTEMP= 5.56675E+02 5.48577E+02 5.19003E+02 5.18284E+02

FH_HP2

HEAT= 2.99649E+06
 SUBCOOL= 5.55000E+00
 AREA= 2.34280E+01
 TTD= 1.87690E+01
 DCTD= 5.76532E+01
 HDP= 1.00000E-02
 CDP = 3.00000E-03 3.00000E-03 3.00000E-03
 AREAS= 1.49895E+01 8.43851E+00 6.67796E-01
 HEATS= 2.99660E+05 2.61804E+06 7.87880E+04
 US = 3.03582E+02 6.37822E+03 1.97282E+03
 LHTDS= 6.53518E+01 4.86421E+01 5.98039E+01
 HTEMP= 6.24388E+02 5.51321E+02 5.37253E+02 5.31703E+02
 CTEMP= 5.18284E+02 5.13970E+02 4.75245E+02 4.74049E+02

FH_LP1

HEAT= 2.69449E+06
 SUBCOOL= 5.55000E+00
 AREA= 2.28340E+01
 TTD= 1.48090E+01
 DCTD= 5.64551E+01
 HDP= 1.00000E-02
 CDP = 3.00000E-03 3.00000E-03 3.00000E-03
 AREAS= 9.51650E+00 1.33195E+01 8.24430E-01
 HEATS= 8.50713E+04 2.52852E+06 8.03979E+04
 US = 1.40454E+02 4.13152E+03 1.67761E+03
 LHTDS= 6.36461E+01 4.59484E+01 5.84916E+01
 HTEMP= 5.17469E+02 4.42722E+02 4.25117E+02 4.19567E+02
 CTEMP= 4.10309E+02 4.08833E+02 3.64540E+02 3.63112E+02

FH_LP2

HEAT= 1.90369E+06
 SUBCOOL= 5.55000E+00
 AREA= 2.38097E+01
 TTD= 9.75205E+00
 DCTD= 3.79210E+01
 HDP= 1.00000E-02
 CDP = 3.00000E-03 3.00000E-03 3.00000E-03
 AREAS= 0.00000E+00 2.33097E+01 1.41857E+00
 HEATS= 0.00000E+00 1.81283E+06 9.08588E+04
 US = 3.40000E+02 3.45484E+03 1.60700E+03
 LHTDS= 0.00000E+00 2.20382E+01 3.98565E+01
 HTEMP= 3.72864E+02 3.72864E+02 3.72864E+02 3.67314E+02
 CTEMP= 3.63112E+02 3.63112E+02 3.31007E+02 3.29393E+02

PUMP_SC

EXIT PRESSURE = 1.50000E+01
 EFFICIENCY = 9.00000E-01

HX_ECON

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 15.45 13.48 KG/S
 INLET TEMPERATURES = 584.49 327.61 K
 AVERAGE TEMPERATURES = 581.64 328.50 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 2.53129E+02 K
 HEAT TRANSFERRED = 1.00000E+05 W
 HEAT TRANSFER SURFACE AREA = 3.95055E+02 SQ-M
 HEAT FLUX = 2.53129E+02 W/SQ-M
 SURFACE TEMPERATURES = 331.36 331.36 K

DEAR_1
 QUAL = 6.4744E-10

PUMP_FW

EXIT PRESSURE = 1.80000E+02
 EFFICIENCY = 9.00000E-01

PUMP_BFP

EXIT PRESSURE = 1.81818E+02
 EFFICIENCY = 9.00000E-01

HX_BOIL

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 15.45 75.00 KG/S
 INLET TEMPERATURES = 2331.78 621.71 K
 AVERAGE TEMPERATURES = 1998.43 626.43 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 1.34534E+03 K
 HEAT TRANSFERRED = 1.86251E+07 W
 HEAT TRANSFER SURFACE AREA = 1.38441E+04 SQ-M
 HEAT FLUX = 1.34534E+03 W/SQ-M
 SURFACE TEMPERATURES = 986.43 986.43 K

IN_AIR

ID=GAS
 TEMP = 2.98150E+02
 PRES = 1.00000E+00
 VEL = 0.00000E+00
 ENTH = -9.91020E+00
 MASS = 1.40000E+01

CP_AIR

MODE = DESIGN
 EXIT PRES = 1.15000E+00
 EFFICIENCY = 8.80000E-01
 MASS FACTOR = 4.04731E-02
 H FACTOR = 1.00000E+00
 PRESSURE RATIO = 1.15000E+00

HX_AIR

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 15.45 14.00 KG/S
 INLET TEMPERATURES = 734.40 311.91 K
 AVERAGE TEMPERATURES = 659.44 405.96 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/H
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 2.53006E+02 K
 HEAT TRANSFERRED = 2.68833E+06 W
 HEAT TRANSFER SURFACE AREA = 1.06255E+04 SQ-M
 HEAT FLUX = 2.53006E+02 W/SQ-M
 SURFACE TEMPERATURES = 481.39 481.39 K

INF_COAL

FUEL HHV= 2.07430E+07
 FUEL MASS= 2.00000E+00

FUEL WEIGHT FRACTIONS

CARBON	HYDROGEN	OXYGEN	NITROGEN	SULFUR	CHLORINE	WATER	ASH
0.521300	0.060000	0.315200	0.007900	0.008500	0.000000	0.227000	0.087100

DRY_1

FUEL HHV= 2.54927E+07
 FUEL MASS= 1.62737E+00
 H2O_DET= 5.00000E-02
 H2O REMOVED= 3.72632E-01
 HEAT_REQUIRED= 9.17862E+05

FUEL WEIGHT FRACTIONS

CARBON	HYDROGEN	OXYGEN	NITROGEN	SULFUR	CHLORINE	WATER	ASH
0.640666	0.073739	0.387374	0.009709	0.010446	0.000000	0.050000	0.107044

CB_1

FUEL MASS BURNED = 1.45317E+00
 FUEL HHV = 2.85407E+07
 FUEL HEAT OF FORM AS BURNED = -4.38836E+06
 HEAT LOSS FRACTION = 0.00000E+00
 STOICHIOMETRY = 1.09462E+00
 CARBON BURNOUT = 1.00000E+00
 ASH MASS REMOVED = 1.74200E-01
 ASH MASS IN FUEL = 0.00000E+00
 WATER MASS IN FUEL = 8.13684E-02
 SLURRY CONCENTRATION = 9.44006E-01
 POTASSIUM MASS = 0.00000E+00
 SEED FRACTION = 0.00000E+00

FUEL ELEMENT FRACTIONS (AS BURNED)

ARGON	CARBON	HYDROGEN	POTASSIUM	NITROGEN	OXYGEN	SULFUR	CHLORINE
0.000000	0.717467	0.083844	0.000000	0.010873	0.483539	0.011699	0.000000

OXIDIZER ELEMENT FRACTIONS

ARGON	CARBON	HYDROGEN	POTASSIUM	NITROGEN	OXYGEN	SULFUR	CHLORINE
0.000000	0.000000	0.000000	0.000000	0.756328	0.243672	0.000000	0.000000

GAS ELEMENT FRACTIONS

ARGON	CARBON	HYDROGEN	POTASSIUM	NITROGEN	OXYGEN	SULFUR	CHLORINE
0.000000	0.067468	0.008355	0.000000	0.686227	0.266229	0.001100	0.000000

SK_1

ENERGY REJECTED = 7.75985E+06 W

OUTPUT BY FLOW

FLOW: LIQ_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_H2O	1.800E+02	6.311E+02	0.000E+00	1.891E+06	7.500E+01	2.178E-03	1.418E+08	2.0E-01
SD_1	1.800E+02	6.311E+02	0.000E+00	1.740E+06	6.000E+01	2.178E-03	1.044E+08	0.0E+00
MX_FW	1.768E+02	6.212E+02	0.000E+00	1.642E+06	7.500E+01	1.678E-03	1.232E+08	-1.1E-01
PUMP_BFP	1.818E+02	6.217E+02	0.000E+00	1.643E+06	7.500E+01	1.674E-03	1.232E+08	-1.4E-01
HX_BOIL	1.800E+02	6.311E+02	0.000E+00	1.891E+06	7.500E+01	2.178E-03	1.418E+08	2.0E-01

FLOW: STM_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
SD_1	1.800E+02	6.311E+02	0.000E+00	2.495E+06	1.500E+01	7.250E-03	3.743E+07	1.0E+00
HX_SH	1.782E+02	8.110E+02	0.000E+00	3.380E+06	1.500E+01	1.871E-02	5.070E+07	2.1E+00
ST_HP1	1.000E+02	7.202E+02	0.000E+00	3.231E+06	1.350E+01	2.913E-02	4.361E+07	1.4E+00
ST_HP2	5.000E+01	6.249E+02	0.000E+00	3.071E+06	1.215E+01	5.139E-02	3.732E+07	1.2E+00
HX_RH	4.950E+01	8.110E+02	0.000E+00	3.522E+06	1.215E+01	7.222E-02	4.279E+07	1.4E+00
ST_IP	1.500E+01	6.450E+02	0.000E+00	3.194E+06	1.063E+01	1.912E-01	3.397E+07	1.2E+00
ST_LP1	5.000E+00	5.176E+02	0.000E+00	2.949E+06	1.010E+01	4.627E-01	2.979E+07	1.1E+00
ST_LP2	1.000E+00	3.731E+02	0.000E+00	2.675E+06	9.597E+00	9.064E-01	2.567E+07	1.0E+00
ST_LP3	6.600E-02	3.113E+02	0.000E+00	2.330E+06	9.597E+00	1.006E-02	2.236E+07	9.0E-01
SC_1	6.600E-02	3.113E+02	0.000E+00	1.599E+05	9.597E+00	1.007E-03	1.535E+06	0.0E+00
PUMP_SC	6.600E-02	3.113E+02	0.000E+00	2.275E+05	1.348E+01	1.036E-03	3.068E+06	2.8E-02
PUMP_SC	1.500E+01	3.276E+02	0.000E+00	2.293E+05	1.348E+01	1.014E-03	3.091E+06	-3.2E-01
HX_ECON	1.485E+01	3.294E+02	0.000E+00	2.367E+05	1.348E+01	1.015E-03	3.191E+06	-3.1E-01
FH_LP2	1.476E+01	3.631E+02	0.000E+00	3.779E+05	1.348E+01	1.035E-03	5.095E+06	-2.4E-01
FH_LP1	1.463E+01	4.103E+02	0.000E+00	5.777E+05	1.348E+01	1.076E-03	7.790E+06	-1.4E-01
DEAR_1	1.463E+01	4.709E+02	0.000E+00	8.422E+05	1.500E+01	1.153E-03	1.263E+07	6.5E-10
PUMP_FW	1.800E+02	4.740E+02	0.000E+00	8.636E+05	1.500E+01	1.142E-03	1.295E+07	-1.2E+00
FH_HP2	1.784E+02	5.183E+02	0.000E+00	1.063E+06	1.500E+01	1.217E-03	1.595E+07	-8.7E-01
FH_HP1	1.768E+02	5.567E+02	0.000E+00	1.249E+06	1.500E+01	1.312E-03	1.873E+07	-6.1E-01
MX_FW	1.768E+02	5.567E+02	0.000E+00	1.249E+06	1.500E+01	1.312E-03	1.873E+07	-6.1E-01

FLOW: STM_HP1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_HP1	1.000E+02	7.202E+02	0.000E+00	3.231E+06	1.500E+00	2.913E-02	4.846E+06	1.4E+00
FH_HP1	9.900E+01	5.788E+02	0.000E+00	1.375E+06	1.500E+00	1.424E-03	2.063E+06	0.0E+00
FH_HP2	9.900E+01	5.788E+02	0.000E+00	1.375E+06	1.500E+00	1.424E-03	2.063E+06	0.0E+00

FLOW: STM_HP2

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_HP2	5.000E+01	6.249E+02	0.000E+00	3.071E+06	1.350E+00	5.139E-02	4.146E+06	1.2E+00
FH_HP2	4.950E+01	5.317E+02	0.000E+00	1.127E+06	2.850E+00	1.271E-03	3.212E+06	0.0E+00
FH_LP1	4.950E+01	5.317E+02	0.000E+00	1.127E+06	2.850E+00	1.271E-03	3.212E+06	0.0E+00

FLOW: STM_IP

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_IP	1.500E+01	6.450E+02	0.000E+00	3.194E+06	1.516E+00	1.912E-01	4.843E+06	1.2E+00
DEAR_1	1.500E+01	6.450E+02	0.000E+00	3.194E+06	1.516E+00	1.912E-01	4.843E+06	1.2E+00

FLOW: STM_LP1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_LP1	5.000E+00	5.176E+02	0.000E+00	2.949E+06	5.317E-01	4.627E-01	1.568E+06	1.1E+00
FH_LP1	4.950E+00	4.196E+02	0.000E+00	6.168E+05	3.382E+00	1.087E-03	2.086E+06	0.0E+00
FH_LP2	4.950E+00	4.196E+02	0.000E+00	6.168E+05	3.382E+00	1.087E-03	2.086E+06	0.0E+00

FLOW: STM_LP2

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_LP2	1.000E+00	3.731E+02	0.000E+00	2.675E+06	5.051E-01	9.064E-01	1.351E+06	1.0E+00
FH_LP2	9.900E-01	3.673E+02	0.000E+00	3.945E+05	3.887E+00	1.039E-03	1.533E+06	0.0E+00
MX_SC	9.900E-01	3.673E+02	0.000E+00	3.945E+05	3.887E+00	1.039E-03	1.533E+06	0.0E+00

FLOW: STM_DUM

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_LP3	6.600E-02	3.113E+02	0.000E+00	2.330E+06	0.000E+00	1.006E-02	0.000E+00	9.0E-01
FH_HP1	6.600E-02	3.113E+02	0.000E+00	2.330E+06	0.000E+00	1.006E-02	0.000E+00	9.0E-01

FLOW: AIR_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_AIR	1.000E+00	2.981E+02	0.000E+00	-9.910E+00	1.400E+01	8.468E-01	-1.387E+02	1.0E+00
CP_AIR	1.150E+00	3.119E+02	0.000E+00	1.389E+04	1.400E+01	7.704E-01	1.945E+05	1.0E+00
HX_AIR	1.138E+00	5.000E+02	0.000E+00	2.059E+05	1.400E+01	1.247E+00	2.883E+06	1.0E+00
CB_1	1.138E+00	5.000E+02	0.000E+00	2.059E+05	1.400E+01	1.247E+00	2.883E+06	1.0E+00

FLOW: GAS_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
CB_1	1.138E+00	2.332E+03	0.000E+00	-2.261E+05	1.545E+01	5.772E+00	-3.494E+06	1.0E+00
HX_BOIL	1.127E+00	1.665E+03	0.000E+00	-1.431E+06	1.545E+01	4.109E+00	-2.212E+07	1.0E+00
HX_SH	1.116E+00	1.023E+03	0.000E+00	-2.290E+06	1.545E+01	2.549E+00	-3.539E+07	1.0E+00
HX_RH	1.105E+00	7.344E+02	0.000E+00	-2.644E+06	1.545E+01	1.849E+00	-4.086E+07	1.0E+00
HX_AIR	1.094E+00	5.845E+02	0.000E+00	-2.818E+06	1.545E+01	1.486E+00	-4.355E+07	1.0E+00
HX_ECON	1.083E+00	5.788E+02	0.000E+00	-2.825E+06	1.545E+01	1.487E+00	-4.365E+07	1.0E+00
DRY_1	1.083E+00	7.303E+02	0.000E+00	-2.883E+06	1.583E+01	1.904E+00	-4.562E+07	1.0E+00
SK_1	1.000E+00	2.982E+02	0.000E+00	-3.373E+06	1.583E+01	8.417E-01	-5.338E+07	1.0E+00

COMPOSITION OUTPUT BY FLOW

FLOW: AIR_1

IN_AIR	N2 = 0.78000	O2 = 0.22000
CP_AIR	N2 = 0.78000	O2 = 0.22000
HX_AIR	N2 = 0.78000	O2 = 0.22000
CB_1	N2 = 0.78000	O2 = 0.22000

FLOW: GAS_1

CB_1	CO = 0.01833	CO2= 0.14056	H = 0.00081	H2 = 0.00259	H2O= 0.11133	NO = 0.00542	N2 = 0.69019
	O = 0.00126	OH = 0.00581	O2 = 0.02274	SO2= 0.00097			
HX_BOIL	CO = 0.00008	CO2= 0.16091	H2 = 0.00002	H2O= 0.11866	NO = 0.00072	N2 = 0.70170	O = 0.00001
	OH = 0.00019	O2 = 0.01673	SO2= 0.00098				
HX_SH	CO2= 0.16100	H2O= 0.11879	NO = 0.00001	N2 = 0.70213	O2 = 0.01709	SO2= 0.00098	
HX_RH	CO2= 0.16100	H2O= 0.11879	N2 = 0.70213	O2 = 0.01709	SO2= 0.00098		
HX_AIR	CO2= 0.16100	H2O= 0.11879	N2 = 0.70213	O2 = 0.01709	SO2= 0.00098		
HX_ECON	CO2= 0.15489	H2O= 0.15227	N2 = 0.67546	O2 = 0.01644	SO2= 0.00095		
DRY_1	CO2= 0.15489	H2O= 0.15227	N2 = 0.67546	O2 = 0.01644	SO2= 0.00095		
SK_1	CO2= 0.15489	H2O= 0.15227	N2 = 0.67546	O2 = 0.01644	SO2= 0.00095		

POWER SUMMARY

MODEL	INPUT (W)	PRODUCED (W)	CONSUMED (W)	LOSS (W)
IN_H2O	1.388E+03	0.000E+00	0.000E+00	0.000E+00
ST_HP1	0.000E+00	2.185E+06	0.000E+00	0.000E+00
ST_HP2	0.000E+00	2.098E+06	0.000E+00	0.000E+00
ST_IP	0.000E+00	3.878E+06	0.000E+00	0.000E+00
ST_LP1	0.000E+00	2.547E+06	0.000E+00	0.000E+00
ST_LP2	0.000E+00	2.693E+06	0.000E+00	0.000E+00
ST_LP3	0.000E+00	3.229E+06	0.000E+00	0.000E+00
SC_1	0.000E+00	0.000E+00	0.000E+00	2.032E+07
PUMP_SC	0.000E+00	0.000E+00	2.349E+04	0.000E+00
PUMP_FH	0.000E+00	0.000E+00	3.220E+05	0.000E+00
PUMP_BFP	0.000E+00	0.000E+00	7.132E+04	0.000E+00
IN_AIR	0.000E+00	0.000E+00	0.000E+00	0.000E+00
CP_AIR	0.000E+00	0.000E+00	1.946E+05	0.000E+00
INF_COAL	4.149E+07	0.000E+00	0.000E+00	0.000E+00
CB_1	0.000E+00	0.000E+00	0.000E+00	0.000E+00
SK_1	0.000E+00	0.000E+00	0.000E+00	7.760E+06
SYST_1	4.149E+07	1.663E+07	6.114E+05	2.858E+07
NET	1.602E+07			
AUXILIARY	0.000E+00			
EFFICIENCY	3.862E-01			

SUBSYSTEM: A

CONVERGENCE OF THE INDEPENDENT VARIABLES,
POSSIBLY VERY CLOSE TO THE SOLUTION

OBJECTIVE: 3.42429E+02

VARIABLES

- 1.86251E+07 HX_BOIL.HEAT
- 1.81818E+02 PUMP_BFP.EXIT_PRES
- 1.24777E-01 ST_IP.SR

CONSTRAINTS

- 1.85049E+01 IN_H2O.DH=0.0
- 1.87502E-10 IN_H2O.DP=0.0
- 6.47443E-10 DEAR_1.PARM.QUAL=0.0

APPENDIX E: OPEN-CYCLE MAGNETOHYDRODYNAMIC POWER PLANT

THE POWER OF THE
MACHINE

PROCESS GP_1:IN

PROCESS

```
GAS_O2-> IN_O2
AIR_1-> IN_AIR MX_O2 <-GAS_O2 CP_AIR HX_AIR:C
FUEL_1-> INF_COAL DRY_1 CB_1 <-AIR_1 ->GAS_1
GAS_1-> NZ_1 MG_1:H DF_1
```

SYSTEM A

PROCESS

```
LIQ_1-> IN_H2O SD_1 ->STM_1
STM_1-> HX_SH:C ST_HP1 ->STM_HP1
          ST_HP2 ->STM_HP2
          HX_RH:C ST_IP ->STM_IP
          ST_LP1 ->STM_LP1
          ST_LP2 ->STM_LP2 ST_LP3 ->STM_DUM
          SC_1
STM_HP1-> FH_HP1:H <-STM_DUM
STM_HP2-> FH_HP2:H <-STM_HP1
STM_LP1-> FH_LP1:H <-STM_HP2
STM_LP2-> FH_LP2:H <-STM_LP1
STM_1-> MX_SC <-STM_LP2
          PUMP_SC HX_ECON:C FH_LP2:C FH_LP1:C
          DEAR_1 <-STM_IP
          PUMP_FH FH_HP2:C FH_HP1:C
LIQ_1-> MX_FH <-STM_1
          PUMP_BFP MG_1:C HX_BOIL:C IN_H2O:CYCL
```

```
VARY HX_BOIL.HEAT = * 1E1 25E6
CONS IN_H2O.DN = 0.0
VARY PUMP_BFP.EXIT_PRES = * 100 200
CONS IN_H2O.DP = 0.0
VARY ST_IP.SR = * 0.01 0.20
CONS DEAR_1.PARM.QUAL = 0.0
```

SYSTEM A

PROCESS

```
GAS_1-> HX_BOIL:H HX_SH:H HX_RH:H
          HX_AIR:H HX_ECON:H DRY_1:H SK_1
NULL-> SYST_1 *:*:OUT
```

DATA

```
IN_O2.PARM .ID='GAS'; .T=298.15; .P=1.0; .M=2.0;
          .XO2=1.0;
IN_AIR.PARM .ID='GAS'; .T=298.15; .P=1.0; .M=8.0;
          .XH2=0.78; .XO2=0.22;
IN_H2O.PARM .ID='H2O'; .T=0.0; .P=180.0; .M=75.0;
          .Q=0.20;
INF_COAL.PARM .MASS=2.5; .C=0.5213; .H=0.060; .O=0.3152;
          .N=0.0079; .S=0.0085; .H2O=0.227;
          .ASH=0.0371; .HHV=20.743E6;
SD_1.PARM .QUAL=0.20;
CP_AIR.PARM .EXIT_PRES=6.00; .EFFICIENCY=0.88;
CB_1.PARM .ASH_DET=0.0; .K_FRAC=0.01;
NZ_1.PARM .EFFICIENCY=0.90; .EXIT_VEL=750;
MG_1.PARM .B_FIELD=6.0; .DELTA_LENGTH=1.0; .EXIT_PRES=0.85;
          .FRICTION_COEF=3E-3; .INVERTER_EFF=0.97; .LOAD_FACTOR=0.7;
          .STANTON_NO=2.5E-3; .WALL_TEMP=1800.0; .PRINT=0;
DF_1.PARM .EXIT_VEL=0.0; .PRES_RECOVERY_COEF=0.50;
DRY_1.PARM .H2O_DET=0.05;
HX_BOIL.PARM .HEAT=12E6;
HX_SH.PARM .T_SET(2)=811.;
HX_RH.PARM .T_SET(2)=811.;
HX_AIR.PARM .T_SET(2)=800.;
HX_ECON.PARM .HEAT=1E5;
ST_HP1.PARM .EXIT_PRES=100.; .EFFICIENCY=0.84; .SR=0.10;
ST_HP2.PARM .EXIT_PRES=50.; .EFFICIENCY=0.84; .SR=0.10;
ST_IP.PARM .EXIT_PRES=15.; .EFFICIENCY=0.86; .SR=0.07;
ST_LP1.PARM .EXIT_PRES=5.; .EFFICIENCY=0.87; .SR=0.05;
ST_LP2.PARM .EXIT_PRES=1.; .EFFICIENCY=0.87; .SR=0.05;
```

```

ST_LP3.PARM .EXIT_FRES=0.066; .EFFICIENCY=0.87; .SR=0.0;
SC_1.PARM .EXIT_FRES=0.066;
PUMP_SC.PARM .EXIT_FRES=15.0; .EFFICIENCY=0.90;
PUMP_FN.PARM .EXIT_FRES=180.0; .EFFICIENCY=0.90;
PUMP_BP.PARM .EXIT_FRES=190.0; .EFFICIENCY=0.90;
SYST_1.PARM .POWER_HEAD_PTR=POWER_HEAD_PTR;
          .FLOW_HEAD_PTR=FLOW_HEAD_PTR;

```

```

LOOP: A N= 1 F= 3.7540E+07
X= 1.2000E+07 1.9000E+02 7.0000E-02
C= -6.1270E+03 8.1000E+00 -6.3133E-02

```

```

LOOP: A
S= 4.5277E+05 9.9000E-01 1.1525E+00
MU= 0.00000E+00

```

```

LOOP: A N= 5 F= 3.4243E+02
X= 1.0713E+07 1.8182E+02 1.2478E-01
C= -1.8505E+01 1.8750E-10 6.4744E-10
SCALE TERMINATION, ACTUAL= 1.67041E-09 IN LOOP: A

```

IN_O2

```

ID=GAS
TEMP = 2.98150E+02
PRES = 1.00000E+00
VEL = 0.00000E+00
ENTH = -1.17254E+01
MASS = 2.00000E+00

```

IN_AIR

```

ID=GAS
TEMP = 2.98150E+02
PRES = 1.00000E+00
VEL = 0.00000E+00
ENTH = -9.91020E+00
MASS = 8.00000E+00

```

CP_AIR

```

MODE = DESIGN
EXIT PRES = 6.00000E+00
EFFICIENCY = 8.80000E-01
MASS FACTOR = 2.86271E-02
H FACTOR = 1.00000E+00
PRESSURE RATIO = 6.00000E+00

```

HX_AIR

```

MODE = DESIGN
TYPE = COUNTER
DESIGN MASS FLOW RATES = 11.87 10.00 KG/S
INLET TEMPERATURES = 975.06 519.76 K
AVERAGE TEMPERATURES = 877.31 659.88 K
DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/H
OVERALL HEAT TRANSFER COEF = 1.0000E+00 W/SQ-M K
LOG MEAN TEMP DIFFERENCE = 2.14647E+02 K
HEAT TRANSFERRED = 2.97324E+06 W
HEAT TRANSFER SURFACE AREA = 1.38518E+04 SQ-M
HEAT FLUX = 2.14647E+02 W/SQ-M
SURFACE TEMPERATURES = 760.41 760.41 K

```

INF_COAL

```

FUEL HHV= 2.07430E+07
FUEL MASS= 2.50000E+00

```

FUEL WEIGHT FRACTIONS

CARBON	HYDROGEN	OXYGEN	NITROGEN	SULFUR	CHLORINE	WATER	ASH
0.521300	0.060000	0.315200	0.007900	0.008500	0.000000	0.227000	0.087100

DRY_1

FUEL HHV= 2.54927E+07
 FUEL MASS= 2.03421E+00
 H₂O DET= 5.00000E-02
 H₂O REMOVED= 4.65789E-01
 HEAT_REQUIRED= 1.14733E+06

FUEL WEIGHT FRACTIONS

CARBON	HYDROGEN	OXYGEN	NITROGEN	SULFUR	CHLORINE	WATER	ASH
0.640666	0.073739	0.383734	0.009709	0.010446	0.000000	0.050000	0.107044

CB_1

FUEL MASS BURNED = 1.81646E+00
 FUEL HHV = 2.85487E+07
 FUEL HEAT OF FORM AS BURNED = -4.38836E+06
 HEAT LOSS FRACTION = 0.00000E+00
 STOICHIOMETRY = 1.01379E+00
 CARBON BURNOUT = 1.00000E+00
 ASH MASS REMOVED = 2.17750E-01
 ASH MASS IN FUEL = 0.00000E+00
 WATER MASS IN FUEL = 1.01711E-01
 SLURRY CONCENTRATION = 9.44006E-01
 POTASSIUM MASS = 5.76982E-02
 SEED FRACTION = 1.00000E-02

FUEL ELEMENT FRACTIONS (AS BURNED)

ARGON	CARBON	HYDROGEN	POTASSIUM	NITROGEN	OXYGEN	SULFUR	CHLORINE
0.000000	0.717467	0.088344	0.000000	0.010873	0.483539	0.011699	0.000000

OXIDIZER ELEMENT FRACTIONS

ARGON	CARBON	HYDROGEN	POTASSIUM	NITROGEN	OXYGEN	SULFUR	CHLORINE
0.000000	0.000000	0.000000	0.000000	0.605062	0.394938	0.000000	0.000000

GAS ELEMENT FRACTIONS

ARGON	CARBON	HYDROGEN	POTASSIUM	NITROGEN	OXYGEN	SULFUR	CHLORINE
0.000000	0.109755	0.013591	0.004859	0.511225	0.406573	0.001790	0.000000

NB_1

EFFICIENCY = 9.00000E-01
 EXIT VELOCITY = 7.50000E+02

MG_1

STANTON NO. = 2.50000E-03
 FRICTION COEFFICIENT = 3.00000E-03
 EXIT PRESSURE = 8.50000E-01
 WALL TEMPERATURE = 1.80000E+03
 LOAD FACTOR = 7.00000E-01
 FARADAY FIELD = 3.15000E+03
 FARADAY CURRENT = 7.16795E+03
 HALL FIELD = 4.53359E+03
 MAGNETIC FIELD INTENSITY = 6.00000E+00
 PERCENTAGE HEAT LOSS = 8.27536E+01
 PERCENTAGE PRESSURE LOSS = 6.92281E+00
 MAXIMUM HALL PARAMETER = 7.49928E+00
 AVERAGE CONDUCTIVITY = 5.30959E+00
 INLET MACH NO. = 7.96541E-01
 OUTLET MACH NO. = 8.77318E-01
 POWER DENSITY = 1.25017E+07
 FLOW RATIO (L/D) = 1.94868E+01
 INLET AREA = 3.06991E-02
 OUTLET AREA = 1.45865E-01
 CHANNEL LENGTH = 7.00000E+00

DF_1

PRESSURE RECOVERY COEFFICIENT = 5.00000E-01
EXIT VELOCITY = 0.00000E+00

IN_H2O

ID=H2O
TEMP = 6.31148E+02
PRES = 1.00000E+02
VEL = 0.00000E+00
ENTH = 1.89132E+06
MASS = 7.50000E+01

SD_1

QUALITY = 2.00000E-01

HX_SH

MODE = DESIGN
TYPE = COUNTER
DESIGN MASS FLOW RATES = 11.87 15.00 KG/S
INLET TEMPERATURES = 2015.08 631.15 K
AVERAGE TEMPERATURES = 1665.14 721.07 K
DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
LOG MEAN TEMP DIFFERENCE = 9.19689E+02 K
HEAT TRANSFERRED = 1.32690E+07 W
HEAT TRANSFER SURFACE AREA = 1.44277E+04 SQ-M
HEAT FLUX = 9.19689E+02 W/SQ-M
SURFACE TEMPERATURES = 1095.39 1095.39 K

ST_HP1

MODE = DESIGN
TURBINE EFFICIENCY = 8.40000E-01
MECHANICAL EFFICIENCY = 9.75000E-01
POWER PRODUCED = 2.18482E+06
FLOW FACTOR = 2.36590E-05
DESIGN MASS FLOW RATE = 1.50000E+01
SPLIT RATIO = 1.00000E-01
VOL FLOW RATE = 4.36901E-01
EXHAUST LOSS = 0.00000E+00

ST_HP2

MODE = DESIGN
TURBINE EFFICIENCY = 8.40000E-01
MECHANICAL EFFICIENCY = 9.75000E-01
POWER PRODUCED = 2.09772E+06
FLOW FACTOR = 3.57563E-05
DESIGN MASS FLOW RATE = 1.35000E+01
SPLIT RATIO = 1.00000E-01
VOL FLOW RATE = 6.93773E-01
EXHAUST LOSS = 0.00000E+00

HX_RH

MODE = DESIGN
TYPE = COUNTER
DESIGN MASS FLOW RATES = 11.87 12.15 KG/S
INLET TEMPERATURES = 1315.19 624.86 K
AVERAGE TEMPERATURES = 1145.12 717.93 K
DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
LOG MEAN TEMP DIFFERENCE = 4.22530E+02 K
HEAT TRANSFERRED = 5.47389E+06 W
HEAT TRANSFER SURFACE AREA = 1.29551E+04 SQ-M
HEAT FLUX = 4.22530E+02 W/SQ-M
SURFACE TEMPERATURES = 892.66 892.66 K

ST_IP

MODE = DESIGN
 TURBINE EFFICIENCY = 8.60000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 3.87820E+06
 FLOW FACTOR = 6.83866E-05
 DESIGN MASS FLOW RATE = 1.21500E+01
 SPLIT RATIO = 1.24777E-01
 VOL FLOW RATE = 2.32286E+00
 EXHAUST LOSS = 0.00000E+00

ST_LP1

MODE = DESIGN
 TURBINE EFFICIENCY = 8.70000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 2.54706E+06
 FLOW FACTOR = 1.77688E-04
 DESIGN MASS FLOW RATE = 1.06340E+01
 SPLIT RATIO = 5.00000E-02
 VOL FLOW RATE = 4.91983E+00
 EXHAUST LOSS = 0.00000E+00

ST_LP2

MODE = DESIGN
 TURBINE EFFICIENCY = 8.70000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 2.69830E+06
 FLOW FACTOR = 4.53640E-04
 DESIGN MASS FLOW RATE = 1.01023E+01
 SPLIT RATIO = 5.00000E-02
 VOL FLOW RATE = 9.15643E+00
 EXHAUST LOSS = 0.00000E+00

ST_LP3

MODE = DESIGN
 TURBINE EFFICIENCY = 8.70000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 3.22837E+06
 FLOW FACTOR = 1.82963E-03
 DESIGN MASS FLOW RATE = 9.59715E+00
 SPLIT RATIO = 0.00000E+00
 VOL FLOW RATE = 9.65782E-02
 EXHAUST LOSS = 0.00000E+00

SC_1

EXIT PRESSURE = 6.60000E-02

FH_HP1

HEAT= 2.78348E+06
 SUPCOOL= 5.55000E+00
 AREA= 1.94182E+01
 TTD= 2.76867E+01
 DCTD= 6.05282E+01
 HDP= 1.00000E-02
 COP = 3.00000E-03 3.00000E-03 3.00000E-03
 AREAS= 1.43913E+01 5.02755E+00 4.18492E-01
 HEATS= 6.08551E+05 2.12486E+06 5.00694E+04
 US = 4.32182E+02 7.17251E+03 1.90172E+03
 LMTDS= 9.78430E+01 5.89255E+01 6.29126E+01
 HTEMP= 7.19632E+02 6.01506E+02 5.84362E+02 5.78312E+02
 CTEMP= 5.56675E+02 5.48577E+02 5.19003E+02 5.18284E+02

FH_HP2

HEAT= 2.99649E+06
 SUBCOOL= 5.55000E+00
 AREA= 2.34280E+01
 TTD= 1.89690E+01
 DCTD= 5.76532E+01
 HDP= 1.00000E-02
 CDP = 3.00000E-03 3.00000E-03 3.00000E-03
 AREAS= 1.49295E+01 8.43851E+00 6.67796E-01
 HEATS= 2.99660E+05 2.61804E+06 7.87880E+04
 US = 3.03582E+02 6.37222E+03 1.97282E+03
 LMTDS= 6.50512E+01 4.86421E+01 5.98039E+01
 HTEMP= 6.24322E+02 5.51321E+02 5.37253E+02 5.31703E+02
 CTEMP= 5.18284E+02 5.13970E+02 4.75245E+02 4.74049E+02

FH_LP1

HEAT= 2.69449E+06
 SUBCOOL= 5.55000E+00
 AREA= 2.28360E+01
 TTD= 1.48030E+01
 DCTD= 5.64551E+01
 HDP= 1.00000E-02
 CDP = 3.00000E-03 3.00000E-03 3.00000E-03
 AREAS= 9.51650E+00 1.33195E+01 8.24430E-01
 HEATS= 8.50713E+04 2.52852E+06 8.08979E+04
 US = 1.40454E+02 4.13152E+03 1.67761E+03
 LMTDS= 6.36461E+01 4.59434E+01 5.84916E+01
 HTEMP= 5.17469E+02 4.42722E+02 4.25117E+02 4.19567E+02
 CTEMP= 4.10309E+02 4.08833E+02 3.64540E+02 3.63112E+02

FH_LP2

HEAT= 1.90369E+06
 SUBCOOL= 5.55000E+00
 AREA= 2.38097E+01
 TTD= 9.75205E+00
 DCTD= 3.79210E+01
 HDP= 1.00000E-02
 CDP = 3.00000E-03 3.00000E-03 3.00000E-03
 AREAS= 0.00000E+00 2.38097E+01 1.41857E+00
 HEATS= 0.00000E+00 1.81283E+06 9.08538E+04
 US = 3.40000E+02 3.45484E+03 1.60700E+03
 LMTDS= 0.00000E+00 2.20382E+01 3.98555E+01
 HTEMP= 3.72864E+02 3.72864E+02 3.72864E+02 3.67314E+02
 CTEMP= 3.63112E+02 3.63112E+02 3.31007E+02 3.29393E+02

PUMP_SC

EXIT PRESSURE = 1.50000E+01
 EFFICIENCY = 9.00000E-01

HX_ECON

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 11.87 13.48 KG/S
 INLET TEMPERATURES = 779.56 327.61 K
 AVERAGE TEMPERATURES = 776.18 328.50 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 4.47676E+02 K
 HEAT TRANSFERRED = 1.00000E+05 W
 HEAT TRANSFER SURFACE AREA = 2.23376E+02 SQ-M
 HEAT FLUX = 4.47676E+02 W/SQ-M
 SURFACE TEMPERATURES = 331.89 331.89 K

DEAR_1

QUAL= 6.4744E-10

PUMP_FH

EXIT PRESSURE = 1.80000E+02
 EFFICIENCY = 9.00000E-01

PUMP_BFP

EXIT PRESSURE = 1.81818E+02
 EFFICIENCY = 9.00000E-01

HX_BOIL

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 11.87 75.00 KG/S
 INLET TEMPERATURES = 2342.85 631.98 K
 AVERAGE TEMPERATURES = 2178.97 631.56 K
 DESIGN THERMAL RESISTIVITIES = 1.00000E+00 0.00000E+00 0.00000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LCG MEAN TEMP DIFFERENCE = 1.54157E+03 K
 HEAT TRANSFERRED = 1.07128E+07 W
 HEAT TRANSFER SURFACE AREA = 6.94930E+03 SQ-M
 HEAT FLUX = 1.54157E+03 W/SQ-M
 SURFACE TEMPERATURES = 801.28 801.28 K

SK_1

ENERGY REJECTED = 9.52601E+06 W

OUTPUT BY FLOW

FLOW: GAS_O2

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_O2	1.000E+00	2.981E+02	0.000E+00	-1.173E+01	2.000E+00	7.646E-01	-2.345E+01	1.0E+00
MX_O2	1.000E+00	2.981E+02	0.000E+00	-1.173E+01	2.000E+00	7.646E-01	-2.345E+01	1.0E+00

FLOW: AIR_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_AIR	1.000E+00	2.981E+02	0.000E+00	-9.910E+00	8.000E+00	8.468E-01	-7.928E+01	1.0E+00
MX_O2	1.000E+00	2.981E+02	0.000E+00	-1.027E+01	1.000E+01	8.304E-01	-1.027E+02	1.0E+00
CP_AIR	6.000E+00	5.198E+02	0.000E+00	2.230E+05	1.000E+01	2.413E-01	2.230E+06	1.0E+00
HX_AIR	5.940E+00	8.000E+02	0.000E+00	5.204E+05	1.000E+01	3.751E-01	5.204E+06	1.0E+00
CB_1	5.940E+00	8.000E+02	0.000E+00	5.204E+05	1.000E+01	3.751E-01	5.204E+06	1.0E+00

FLOW: GAS_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
CB_1	5.940E+00	2.826E+03	0.000E+00	-2.331E+05	1.187E+01	1.369E+00	-2.768E+06	1.0E+00
NZ_1	4.029E+00	2.735E+03	7.500E+02	-5.143E+05	1.187E+01	1.939E+00	-2.768E+06	1.0E+00
MG_1	6.524E-01	2.256E+03	7.500E+02	-2.011E+06	1.187E+01	9.213E+00	-2.054E+07	1.0E+00
DF_1	8.292E-01	2.343E+03	0.000E+00	-1.730E+06	1.187E+01	7.816E+00	-2.054E+07	1.0E+00
HX_BOIL	8.209E-01	2.015E+03	0.000E+00	-2.632E+06	1.187E+01	6.642E+00	-3.125E+07	1.0E+00
HX_SH	8.127E-01	1.315E+03	0.000E+00	-3.749E+06	1.187E+01	4.354E+00	-4.452E+07	1.0E+00
HX_RH	8.046E-01	9.751E+02	0.000E+00	-4.210E+06	1.187E+01	3.261E+00	-4.999E+07	1.0E+00
HX_AIR	7.966E-01	7.796E+02	0.000E+00	-4.461E+06	1.187E+01	2.333E+00	-5.297E+07	1.0E+00
HX_ECON	7.886E-01	7.728E+02	0.000E+00	-4.469E+06	1.187E+01	2.637E+00	-5.307E+07	1.0E+00
DRY_1	7.886E-01	9.330E+02	0.000E+00	-4.562E+06	1.234E+01	3.267E+00	-5.629E+07	1.0E+00
SK_1	1.000E+00	2.982E+02	0.000E+00	-5.334E+06	1.234E+01	8.232E-01	-6.582E+07	1.0E+00

FLOW: LIQ_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_H2O	1.800E+02	6.311E+02	0.000E+00	1.891E+06	7.500E+01	2.178E-03	1.418E+08	2.0E-01
SD_1	1.800E+02	6.311E+02	0.000E+00	1.740E+06	6.000E+01	2.178E-03	1.044E+08	0.0E+00
MX_FW	1.768E+02	6.212E+02	0.000E+00	1.642E+06	7.500E+01	1.674E-03	1.232E+08	-1.1E-01
PUMP_BFP	1.818E+02	6.217E+02	0.000E+00	1.643E+06	7.500E+01	1.674E-03	1.232E+08	-1.4E-01
MG_1	1.818E+02	6.320E+02	0.000E+00	1.748E+06	7.500E+01	1.869E-03	1.311E+08	2.7E-04
HX_BOIL	1.800E+02	6.311E+02	0.000E+00	1.891E+06	7.500E+01	2.178E-03	1.418E+08	2.0E-01

FLOW: STM_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
SD_1	1.800E+02	6.311E+02	0.000E+00	2.495E+06	1.500E+01	7.250E-03	3.743E+07	1.0E+00
HX_SH	1.782E+02	8.110E+02	0.000E+00	3.380E+06	1.500E+01	1.817E-02	5.070E+07	2.1E+00
ST_HP1	1.000E+02	7.202E+02	0.000E+00	3.231E+06	1.350E+01	2.913E-02	4.361E+07	1.4E+00
ST_HP2	5.000E+01	6.249E+02	0.000E+00	3.071E+06	1.215E+01	5.139E-02	3.732E+07	1.2E+00
HX_RH	4.950E+01	8.110E+02	0.000E+00	3.522E+06	1.215E+01	7.222E-02	4.279E+07	1.4E+00
ST_IP	1.500E+01	6.450E+02	0.000E+00	3.194E+06	1.063E+01	1.912E-01	3.397E+07	1.2E+00
ST_LP1	5.000E+00	5.176E+02	0.000E+00	2.949E+06	1.010E+01	4.627E-01	2.979E+07	1.1E+00
ST_LP2	1.000E+00	3.731E+02	0.000E+00	2.675E+06	9.597E+00	9.064E-01	2.567E+07	1.0E+00
ST_LP3	6.600E-02	3.113E+02	0.000E+00	2.330E+06	9.597E+00	1.006E-02	2.236E+07	9.0E-01
SC_1	6.600E-02	3.113E+02	0.000E+00	1.599E+05	9.597E+00	1.007E-03	1.535E+06	0.0E+00
HX_SC	6.600E-02	3.113E+02	0.000E+00	2.275E+05	1.348E+01	1.036E-03	3.068E+06	2.8E-02
PUMP_SC	1.500E+01	3.276E+02	0.000E+00	2.293E+05	1.348E+01	1.014E-03	3.091E+06	-3.2E-01
HX_ECON	1.485E+01	3.294E+02	0.000E+00	2.367E+05	1.348E+01	1.015E-03	3.191E+06	-3.1E-01
FH_LP2	1.476E+01	3.312E+02	0.000E+00	3.779E+05	1.348E+01	1.035E-03	5.095E+06	-2.4E-01
FH_LP1	1.463E+01	4.103E+02	0.000E+00	5.777E+05	1.348E+01	7.076E-03	7.790E+06	-1.4E-01
DEAR_1	1.463E+01	4.709E+02	0.000E+00	8.422E+05	1.500E+01	1.153E-03	1.263E+07	1.5E-10
PUMP_FW	1.800E+02	4.740E+02	0.000E+00	8.636E+05	1.500E+01	1.142E-03	1.295E+07	-1.2E+00
FH_HP2	1.784E+02	5.183E+02	0.000E+00	1.063E+06	1.500E+01	1.217E-03	1.595E+07	-8.7E-01
FH_HP1	1.768E+02	5.567E+02	0.000E+00	1.249E+06	1.500E+01	1.312E-03	1.873E+07	-6.1E-01
MX_FW	1.768E+02	5.567E+02	0.000E+00	1.249E+06	1.500E+01	1.312E-03	1.873E+07	-6.1E-01

FLOW: STM_HP1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_HP1	1.000E+02	7.202E+02	0.000E+00	3.231E+06	1.500E+00	2.913E-02	4.846E+06	1.4E+00
FH_HP1	9.900E+01	5.788E+02	0.000E+00	1.375E+06	1.500E+00	1.424E-03	2.063E+06	0.0E+00
FH_HP2	9.900E+01	5.788E+02	0.000E+00	1.375E+06	1.500E+00	1.424E-03	2.063E+06	0.0E+00

FLOW: STM_HP2

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_HP2	5.000E+01	6.249E+02	0.000E+00	3.071E+06	1.350E+00	5.139E-02	4.166E+06	1.2E+00
FH_HP2	4.950E+01	5.317E+02	0.000E+00	1.127E+06	2.850E+00	1.271E-03	3.212E+06	0.0E+00
FH_LP1	4.950E+01	5.317E+02	0.000E+00	1.127E+06	2.850E+00	1.271E-03	3.212E+06	0.0E+00

FLOW: STM_IP

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_IP	1.500E+01	6.450E+02	0.000E+00	3.194E+06	1.516E+00	1.912E-01	4.843E+06	1.2E+00
DEAR_1	1.500E+01	6.450E+02	0.000E+00	3.194E+06	1.516E+00	1.912E-01	4.843E+06	1.2E+00

FLOW: STM_LP1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_LP1	5.000E+00	5.176E+02	0.000E+00	2.949E+06	5.317E-01	4.627E-01	1.568E+06	1.1E+00
FH_LP1	4.950E+00	4.196E+02	0.000E+00	6.168E+05	3.382E+00	1.087E-03	2.086E+06	0.0E+00
FH_LP2	4.950E+00	4.196E+02	0.000E+00	6.168E+05	3.382E+00	1.087E-03	2.086E+06	0.0E+00

FLOW: STM_LP2

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_LP2	1.000E+00	3.731E+02	0.000E+00	2.675E+06	5.051E-01	9.064E-01	1.351E+06	1.0E+00
FH_LP2	9.900E-01	3.673E+02	0.000E+00	3.945E+05	3.887E+00	1.039E-03	1.533E+06	0.0E+00
HX_SC	9.900E-01	3.673E+02	0.000E+00	3.945E+05	3.887E+00	1.039E-03	1.533E+06	0.0E+00

FLOW: STM_DUM

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_LP3	6.600E-02	3.113E+02	0.000E+00	2.330E+06	0.000E+00	1.006E-02	0.000E+00	9.0E-01
FH_HP1	6.600E-02	3.113E+02	0.000E+00	2.330E+06	0.000E+00	1.006E-02	0.000E+00	9.0E-01

COMPOSITION OUTPUT BY FLOW

FLOW: GAS_O2

IN_O2	O2 = 1.00000
HX_O2	O2 = 1.00000

FLOW: AIR_1

IN_AIR	N2 = 0.78000	O2 = 0.22000
HX_O2	N2 = 0.63636	O2 = 0.36364
CP_AIR	N2 = 0.63636	O2 = 0.36364
HX_AIR	N2 = 0.63636	O2 = 0.36363
CB_1	N2 = 0.63636	O2 = 0.36363

FLOW: GAS_1

CB_1	CO = 0.08649	CO2= 0.16230	H = 0.00612	H2 = 0.01214	H2O= 0.15620	K = 0.00176	KOH= 0.00162
NO	NO = 0.01339	N2 = 0.49018	O = 0.00729	OH = 0.02269	O2 = 0.03828	SO2= 0.00152	
NZ_1	CO = 0.07845	CO2= 0.17212	H = 0.00513	H2 = 0.01099	H2O= 0.16058	K = 0.00179	KOH= 0.00161
NO	NO = 0.01145	N2 = 0.49470	O = 0.00598	OH = 0.01984	O2 = 0.03580	SO2= 0.00153	
HG_1	CO = 0.02921	CO2= 0.23126	H = 0.00095	H2 = 0.00431	H2O= 0.18340	K = 0.00160	KOH= 0.00195
NO	NO = 0.00346	N2 = 0.51847	O = 0.00093	OH = 0.00604	O2 = 0.01684	SO2= 0.00159	
DF_1	CO = 0.03868	CO2= 0.21999	H = 0.00152	H2 = 0.00560	H2O= 0.17947	K = 0.00174	KOH= 0.00178
NO	NO = 0.00459	N2 = 0.51430	O = 0.00155	OH = 0.00826	O2 = 0.02094	SO2= 0.00158	
HX_BOIL	CO = 0.00790	CO2= 0.25655	H = 0.00011	H2 = 0.00129	H2O= 0.19150	K = 0.00073	KOH= 0.00287
NO	NO = 0.00126	N2 = 0.52751	O = 0.00011	OH = 0.00167	O2 = 0.00690	SO2= 0.00162	
HX_SH	CO2= 0.26596	H2O= 0.19441	KOH= 0.00361	NO = 0.00005	N2 = 0.53114	OH = 0.00001	O2 = 0.00319
SO2= 0.00162							
CO2= 0.26596	H2O= 0.19442	KOH= 0.00362	H2 = 0.53116	O2 = 0.00321	SO2= 0.00162		
CO2= 0.26596	H2O= 0.19442	KOH= 0.00362	H2 = 0.53117	O2 = 0.00321	SO2= 0.00162		
CO2= 0.26596	H2O= 0.19442	KOH= 0.00362	H2 = 0.53117	O2 = 0.00321	SO2= 0.00162		
CO2= 0.24940	H2O= 0.24458	KOH= 0.00339	H2 = 0.49809	O2 = 0.00301	SO2= 0.00152		
CO2= 0.24940	H2O= 0.24458	KOH= 0.00339	H2 = 0.49809	O2 = 0.00301	SO2= 0.00152		

POWER SUMMARY

MODEL	INPUT (W)	PRODUCED (W)	CONSUMED (W)	LOSS (W)
IN_O2	0.000E+00	0.000E+00	0.000E+00	0.000E+00
IN_AIR	0.000E+00	0.000E+00	0.000E+00	0.000E+00
CP_AIR	0.000E+00	0.000E+00	2.231E+06	0.000E+00
INF_CCAL	5.135E+07	0.000E+00	0.000E+00	0.000E+00
CB_1	0.000E+00	0.000E+00	0.000E+00	0.000E+00
MG_1	0.000E+00	9.561E+06	0.000E+00	0.000E+00
IN_H2O	1.393E+03	0.000E+00	0.000E+00	0.000E+00
ST_HP1	0.000E+00	2.125E+06	0.000E+00	0.000E+00
ST_HP2	0.000E+00	2.098E+06	0.000E+00	0.000E+00
ST_IP	0.000E+00	3.878E+06	0.000E+00	0.000E+00
ST_LP1	0.000E+00	2.547E+06	0.000E+00	0.000E+00
ST_LP2	0.000E+00	2.693E+06	0.000E+00	0.000E+00
ST_LP3	0.000E+00	3.229E+06	0.000E+00	0.000E+00
SC_1	0.000E+00	0.000E+00	0.000E+00	2.082E+07
PUMP_SC	0.000E+00	0.000E+00	2.349E+04	0.000E+00
PUMP_FW	0.000E+00	0.000E+00	3.220E+05	0.000E+00
PUMP_BFP	0.000E+00	0.000E+00	7.132E+04	0.000E+00
SK_1	0.000E+00	0.000E+00	0.000E+00	9.526E+06
SYST_1	5.186E+07	2.620E+07	2.647E+06	3.035E+07
NET	2.355E+07			
AUXILIARY	0.000E+00			
EFFICIENCY	4.541E-01			

SUBSYSTEM: A

CONVERGENCE OF THE INDEPENDENT VARIABLES,
POSSIBLY VERY CLOSE TO THE SOLUTION

OBJECTIVE: 3.42431E+02

VARIABLES

1	1.07128E+07	HX_BOIL.HEAT
2	1.81315E+02	PUMP_BFP.EXIT_PRES
3	1.24777E-01	ST_IP.SR

CONSTRAINTS

1	-1.85049E+01	IN_H2O.DH=0.0
2	1.87502E-10	IN_H2O.DP=0.0
3	6.47443E-10	DEAR_1.PARH.QUAL=0.0

APPENDIX F: SOLID-OXIDE FUEL-CELL SYSTEM


```

PROCESS
  GP_0:IN
PLI GASFRZ='1'B;
PROCESS
  STM_MIX-> IN_MSTM CP_STM
  GAS_AN-> IN_GAS CP_GAS HX_1:C HX_STM <-STM_MIX HX_A:C
  AIR_1-> IN_AIR CP_AIR1 HT_INTER CP_AIR2 HX_C:C
PLI GASFRZ='0'B;
PROCESS
  GAS_AN-> AIR_1-> SOFC_1
  GAS_AN-> MX_BURN <-AIR_1 SP_BURN ->AIR_1
PLI GASFRZ='1'B;
PROCESS
  AIR_1-> HX_C:H
  GAS_AN-> HX_A:H MX_AIR <-AIR_1
PROCESS
  GAS_AN-> HX_FB:H GT_1 HX_ST:H SK_1
SYSEB A
VARY IN_STM.PARM.M = * 1.0 10.0
VARY PUMP_SC.EXIT_PRES = * 140 170
CONS IN_STM.DH=0.0
CONS IN_STM.DP=0.0
PROCESS
  STM_1-> IN_STM HX_1:H ST_1 ->STM_DUM
  SC_1 PUMP_SC HX_ST:C HX_FB:C IN_STM:CYCL
SYSEND A
PLI SOFC_1.POWER.PRODUCED = 0.96*SOFC_1.POWER.PRODUCED;
  IN_GAS.POWER.INPUT = IN_GAS.PARM.H*55.529E6;
PROCESS
  NULL-> SYST_1 *_:OUT

DATA
IN_GAS.PARM .T=298.0; .P=1.0; .M=1.0;
  .COMP.XCH4=1.000;
IN_STM.PARM .ID='H2O'; .T=823.0; .P=150.0;
  .M=5.0; .COMP.XH2O=1.0;
IN_MSTM.PARM .ID='H2O'; .T=298.15; .P=1.0;
  .M=1.60; .COMP.XH2O=1.0;
IN_AIR.PARM .T=298.0; .P=1.0; .M=30.0;
  .COMP.XO2=0.21; .COMP.XN2=0.79;
HX_1.PARM .T_SET(2)=573.0;
HX_A.PARM .T_SET(2)=1073.0;
HX_C.PARM .T_SET(2)=1073.0;
HX_FB.PARM .T_SET(1)=800.0;
HX_ST.PARM .T_SET(1)=400.0;
CP_AIR1.PARM .EXIT_PRES=3.5; .EFFICIENCY=0.85;
CP_AIR2.PARM .EXIT_PRES=12.0; .EFFICIENCY=0.85;
HT_INTER.PARM .T_SET=318.0;
CP_GAS.PARM .EXIT_PRES=13.0; .EFFICIENCY=0.85;
CP_STM.PARM .EXIT_PRES=12.0; .EFFICIENCY=0.85;
SOFC_1.CELL_CURRENT=1.56863E5;
SOFC_1.DELTA_VOLT=0.180;
SOFC_1.NO_OF_CELLS=230;
SOFC_1.CELL_TEMP=1273.0;
SOFC_1.CELL_VOLTAGE=0;
SP_BURN.PARM.SPLIT_RATIO=0.7;
ST_1.PARM.EXIT_PRES=0.180;
ST_1.PARM.EFFICIENCY=0.82;
SC_1.PARM.EXIT_PRES=0.180;
GT_1.PARM.EFFICIENCY=0.87;
GT_1.PARM.EXIT_PRES=1.0;
PUMP_SC.PARM .EXIT_PRES=150.0;
SYST_1.PARM .POWER_HEAD_PTR=POWER_HEAD_PTR;
  .FLOW_HEAD_PTR=FLOW_HEAD_PTR;

```

LOOP: A N= 1 F= 2.2759E+10
 X= 5.0000E+00 1.5000E+02
 C= -1.5086E+05 -2.9850E+00

LOOP: A
 S= 6.0698E+05 9.8010E-01
 MU= 0.00000E+00

LOOP: A N= 4 F= 6.1692E+07
 X= 4.7520E+00 1.5305E+02
 C= 7.8544E+03 4.4125E-10
 SCALE TERMINATION, ACTUAL= 1.67448E-04 IN LOOP: A

IN_MSTM

ID=H2O
 TEMP = 2.98150E+02
 PRES = 1.00000E+00
 VEL = 0.00000E+00
 ENTH = 1.04976E+05
 MASS = 1.60000E+00

CP_STM

MODE = DESIGN
 EXIT PRES = 1.20000E+01
 EFFICIENCY = 8.50000E-01
 MASS FACTOR = 1.59180E-04
 M FACTOR = 1.00000E+00
 PRESSURE RATIO = 1.20000E+01

IN_GAS

ID=GAS
 TEMP = 2.98000E+02
 PRES = 1.00000E+00
 VEL = 0.00000E+00
 ENTH = -4.66754E+06
 MASS = 1.00000E+00

CP_GAS

MODE = DESIGN
 EXIT PRES = 1.30000E+01
 EFFICIENCY = 8.50000E-01
 MASS FACTOR = 3.87848E-03
 M FACTOR = 1.00000E+00
 PRESSURE RATIO = 1.30000E+01

HX_1

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 4.75 1.00 KG/S
 INLET TEMPERATURES = 823.00 536.94 K
 AVERAGE TEMPERATURES = 818.43 554.97 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 2.63432E+02 K
 HEAT TRANSFERRED = 1.11323E+05 W
 HEAT TRANSFER SURFACE AREA = 4.22586E+02 SQ-M
 HEAT FLUX = 2.63432E+02 W/SQ-M
 SURFACE TEMPERATURES = 559.57 559.57 K

HX_A

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 9.78 2.60 KG/S
 INLET TEMPERATURES = 1637.80 432.74 K
 AVERAGE TEMPERATURES = 1470.80 752.87 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.0000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 7.06908E+02 K
 HEAT TRANSFERRED = 4.55804E+06 W
 HEAT TRANSFER SURFACE AREA = 6.44786E+03 SQ-M
 HEAT FLUX = 7.06908E+02 W/SQ-M
 SURFACE TEMPERATURES = 930.89 930.89 K

IN_AIR

ID=GAS
 TEMP = 2.98000E+02
 PRES = 1.00000E+00
 VEL = 0.00000E+00
 ENTH = -1.61551E+02
 MASS = 3.00000E+01

CP_AIR1

MODE = DESIGN
 EXIT PRES = 3.50000E+00
 EFFICIENCY = 8.50000E-01
 MASS FACTOR = 8.67661E-02
 M FACTOR = 1.00000E+00
 PRESSURE RATIO = 3.50000E+00

HT_INTER

HEAT = -3.95416E+06

CP_AIR2

MODE = DESIGN
 EXIT PRES = 1.20000E+01
 EFFICIENCY = 8.50000E-01
 MASS FACTOR = 2.56037E-02
 M FACTOR = 1.00000E+00
 PRESSURE RATIO = 3.42857E+00

HX_C

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 22.82 30.00 KG/S
 INLET TEMPERATURES = 1637.80 473.84 K
 AVERAGE TEMPERATURES = 1321.17 773.42 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 5.47569E+02 K
 HEAT TRANSFERRED = 1.97448E+07 W
 HEAT TRANSFER SURFACE AREA = 3.60591E+04 SQ-M
 HEAT FLUX = 5.47569E+02 W/SQ-M
 SURFACE TEMPERATURES = 1090.23 1090.23 K

SOFC_1

CELL TEMPERATURE = 1.27300E+03 K
 CELL CURRENT = 1.56863E+05 A
 CELL VOLTAGE = 0.67764 V
 NO OF CELLS = 230
 STACK VOLTAGE = 1.55857E+02 V
 OVERALL ISOTHERMAL HEAT OF REACTION = 4.77208E+07 W
 STACK GROSS POWER = 2.34703E+07 W
 NERNST POTENTIAL AT FUEL CELL EXIT = 8.50413E-01 V
 FUEL UTILIZATION = 7.49853E-01
 OXYGEN UTILIZATION = 4.28455E-01

SP_BURN

SPLIT RATIO = 7.00000E-01
 POWER REQUIRED = 0.00000E+00

HX_FB

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 32.60 4.75 KG/S
 INLET TEMPERATURES = 1095.93 471.10 K
 AVERAGE TEMPERATURES = 947.96 648.49 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 3.31704E+02 K
 HEAT TRANSFERRED = 1.23773E+07 W
 HEAT TRANSFER SURFACE AREA = 3.73143E+04 SQ-M
 HEAT FLUX = 3.31704E+02 W/SQ-M
 SURFACE TEMPERATURES = 764.22 764.22 K

GT_1

MODE = DESIGN
 EXIT PRESSURE = 1.00000E+00
 EFFICIENCY = 8.70000E-01
 MECHANICAL EFFICIENCY = 9.80000E-01
 MASS FACTOR = 1.36216E-02
 M FACTOR = 1.00000E+00
 DESIGN PRESSURE RATIO = 1.16436E+01
 PRESSURE RATIO = 1.16436E+01

HX_ST

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 32.60 4.75 KG/S
 INLET TEMPERATURES = 475.57 332.32 K
 AVERAGE TEMPERATURES = 437.78 401.71 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 3.13512E+01 K
 HEAT TRANSFERRED = 2.79726E+06 W
 HEAT TRANSFER SURFACE AREA = 8.92233E+04 SQ-M
 HEAT FLUX = 3.13512E+01 W/SQ-M
 SURFACE TEMPERATURES = 444.22 444.22 K

SK_1

ENERGY REJECTED = 3.70975E+06 W

IN_STM

ID=H2O
 TEMP = 8.23000E+02
 PRES = 1.50000E+02
 VEL = 0.00000E+00
 ENTH = 3.44609E+06
 MASS = 4.75203E+00

ST_1

MODE = DESIGN
 TURBINE EFFICIENCY = 8.20000E-01
 MECHANICAL EFFICIENCY = 9.75000E-01
 POWER PRODUCED = 4.93119E+06
 FLOW FACTOR = 9.00966E-06
 DESIGN MASS FLOW RATE = 4.75203E+00
 SPLIT RATIO = 0.00000E+00
 VOL FLOW RATE = 4.59741E-02
 EXHAUST LOSS = 0.00000E+00

SC_1

EXIT PRESSURE = 1.80000E-01

PUMP_SC

EXIT PRESSURE = 1.53046E+02
 EFFICIENCY = 9.00000E-01

OUTPUT BY FLOW

FLOW: STM_MIX

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_STM	1.000E+00	2.981E+02	0.000E+00	1.050E+05	1.600E+00	1.003E-03	1.680E+05	-1.4E-01
CP_STM	1.200E+01	2.982E+02	0.000E+00	1.063E+05	1.600E+00	1.002E-03	1.701E+05	-3.5E-01
HX_STM	1.200E+01	2.982E+02	0.000E+00	1.063E+05	1.600E+00	1.002E-03	1.701E+05	-3.5E-01

FLOW: GAS_AN

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_GAS	1.000E+00	2.980E+02	0.000E+00	-4.668E+06	1.000E+00	1.524E+00	-4.668E+06	1.0E+00
CP_GAS	1.300E+01	5.369E+02	0.000E+00	-4.046E+06	1.000E+00	2.113E-01	-4.046E+06	1.0E+00
HX_1	1.287E+01	5.730E+02	0.000E+00	-3.935E+06	1.000E+00	2.277E-01	-3.935E+06	1.0E+00
MX_STM	1.200E+01	4.327E+02	0.000E+00	-9.773E+06	2.600E+00	1.720E-01	-2.541E+07	1.0E+00
HX_A	1.188E+01	1.073E+03	0.000E+00	-8.020E+06	2.600E+00	4.308E-01	-2.085E+07	1.0E+00
SOFC_1	1.188E+01	1.286E+03	0.000E+00	-8.902E+06	5.591E+00	4.380E-01	-4.977E+07	1.0E+00
HX_BURN	1.188E+01	1.638E+03	0.000E+00	-6.191E+05	3.260E+01	4.133E-01	-2.018E+07	1.0E+00
SP_BURN	1.188E+01	1.638E+03	0.000E+00	-6.191E+05	9.780E+00	4.133E-01	-6.055E+06	1.0E+00
HX_A	1.176E+01	1.304E+03	0.000E+00	-1.085E+06	9.780E+00	3.323E-01	-1.061E+07	1.0E+00
HX_AIR	1.176E+01	1.096E+03	0.000E+00	-1.365E+06	3.260E+01	2.794E-01	-4.449E+07	1.0E+00
HX_FB	1.164E+01	8.000E+02	0.000E+00	-1.744E+06	3.260E+01	2.060E-01	-5.686E+07	1.0E+00
GT_1	1.000E+00	4.756E+02	0.000E+00	-2.131E+06	3.260E+01	1.426E+00	-6.946E+07	1.0E+00
HX_ST	9.900E-01	4.000E+02	0.000E+00	-2.217E+06	3.260E+01	1.211E+00	-7.226E+07	1.0E+00
SK_1	1.000E+00	2.982E+02	0.000E+00	-2.330E+06	3.260E+01	8.939E-01	-7.597E+07	1.0E+00

FLOW: AIR_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_AIR	1.000E+00	2.980E+02	0.000E+00	-1.616E+02	3.000E+01	8.476E-01	-4.847E+03	1.0E+00
CP_AIR1	3.500E+00	4.474E+02	0.000E+00	1.519E+05	3.000E+01	3.636E-01	4.556E+06	1.0E+00
HT_INTER	3.500E+00	3.180E+02	0.000E+00	2.007E+04	3.000E+01	2.584E-01	6.021E+05	1.0E+00
CP_AIR2	1.200E+01	4.738E+02	0.000E+00	1.791E+05	3.000E+01	1.123E-01	5.373E+06	1.0E+00
HX_C	1.188E+01	1.073E+03	0.000E+00	8.373E+05	3.000E+01	2.569E-01	2.512E+07	1.0E+00
SOFC_1	1.188E+01	1.286E+03	0.000E+00	1.095E+06	2.701E+01	3.111E-01	2.959E+07	1.0E+00
HX_BURN	1.188E+01	1.286E+03	0.000E+00	1.095E+06	2.701E+01	3.111E-01	2.959E+07	1.0E+00
SP_BURN	1.188E+01	1.638E+03	0.000E+00	-6.191E+05	2.282E+01	4.133E-01	-1.413E+07	1.0E+00
HX_C	1.176E+01	1.005E+03	0.000E+00	-1.484E+06	2.282E+01	2.561E-01	-3.387E+07	1.0E+00
HX_AIR	1.176E+01	1.005E+03	0.000E+00	-1.484E+06	2.282E+01	2.561E-01	-3.387E+07	1.0E+00

FLOW: STM_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_STM	1.500E+02	8.230E+02	0.000E+00	3.446E+06	4.752E+00	2.259E-02	1.638E+07	1.9E+00
HX_1	1.485E+02	8.139E+02	0.000E+00	3.423E+06	4.752E+00	2.247E-02	1.626E+07	1.8E+00
ST_1	1.800E-01	3.312E+02	0.000E+00	2.358E+06	4.752E+00	9.675E-03	1.121E+07	9.0E-01
SC_1	1.800E-01	3.312E+02	0.000E+00	2.432E+05	4.752E+00	1.016E-03	1.156E+06	0.0E+00
PUMP_SC	1.530E+02	3.323E+02	0.000E+00	2.607E+05	4.752E+00	1.010E-03	1.239E+06	-1.4E+00
HX_ST	1.515E+02	4.711E+02	0.000E+00	8.493E+05	4.752E+00	1.140E-03	4.036E+06	-7.9E-01
HX_FB	1.500E+02	8.259E+02	0.000E+00	3.454E+06	4.752E+00	2.270E-02	1.641E+07	1.9E+00

FLOW: STM_DUM

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
ST_1	1.800E-01	3.312E+02	0.000E+00	2.358E+06	0.000E+00	9.675E-03	0.000E+00	9.0E-01

COMPOSITION OUTPUT BY FLOW

FLOW: GAS_AN

IN_GAS	CH4= 1.00000							
CP_GAS	CH4= 1.00000							
HX_1	CH4= 1.00000							
MX_STM	CH4= 0.41240	H2O= 0.58760						
HX_A	CH4= 0.41240	H2O= 0.58760						
SOFC_1	CO = 0.06832	CO2= 0.15768	H2 = 0.15781	H2O= 0.61619				
HX_BURN	CO2= 0.05233	H2O= 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016	O2 = 0.07794		
SP_BURN	CO2= 0.05233	H2O= 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016	O2 = 0.07794		
HX_A	CO2= 0.05233	H2O= 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016	O2 = 0.07794		
HX_AIR	CO2= 0.05233	H2O= 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016	O2 = 0.07794		
HX_FB	CO2= 0.05233	H2O= 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016	O2 = 0.07794		
GT_1	CO2= 0.05233	H2O= 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016	O2 = 0.07794		
HX_ST	CO2= 0.05233	H2O= 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016	O2 = 0.07794		
SK_1	CO2= 0.05233	H2O= 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016	O2 = 0.07794		

FLOW: AIR_1

IN_AIR	N2 = 0.79000	O2 = 0.21000			
CP_AIR1	N2 = 0.79000	O2 = 0.21000			
HT_INTER	N2 = 0.79000	O2 = 0.21000			
CP_AIR2	N2 = 0.79000	O2 = 0.21000			
HX_C	N2 = 0.79000	O2 = 0.21000			
SOFC_1	NO = 0.00033	N2 = 0.86787	O2 = 0.13180		
HX_BURN	NO = 0.00033	N2 = 0.86787	O2 = 0.13180		
SP_BURN	CO2 = 0.05233	H2O = 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016 O2 = 0.07794
HX_C	CO2 = 0.05233	H2O = 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016 O2 = 0.07794
HX_AIR	CO2 = 0.05233	H2O = 0.17915	NO = 0.00139	N2 = 0.68902	OH = 0.00016 O2 = 0.07794

POWER SUMMARY

MODEL	INPUT (W)	PRODUCED (W)	CONSUMED (W)	LOSS (W)
IN_MSTM	0.000E+00	0.000E+00	0.000E+00	0.000E+00
CP_STH	0.000E+00	0.000E+00	2.103E+03	0.000E+00
IN_GAS	5.553E+07	0.000E+00	0.000E+00	0.000E+00
CP_GAS	0.000E+00	0.000E+00	6.215E+05	0.000E+00
IN_AIR	0.000E+00	0.000E+00	0.000E+00	0.000E+00
CP_AIR1	0.000E+00	0.000E+00	4.561E+06	0.000E+00
HT_INTER	0.000E+00	0.000E+00	0.000E+00	3.954E+06
CP_AIR2	0.000E+00	0.000E+00	4.771E+06	0.000E+00
SOFC_1	0.000E+00	2.367E+07	0.000E+00	0.000E+00
ST_1	0.000E+00	1.235E+07	0.000E+00	0.000E+00
SK_1	0.000E+00	0.000E+00	0.000E+00	3.710E+06
IN_STH	-3.732E+04	0.000E+00	0.000E+00	0.000E+00
ST_1	0.000E+00	4.931E+06	0.000E+00	0.000E+00
SC_1	0.000E+00	0.000E+00	0.000E+00	1.005E+07
PUMP_SC	0.000E+00	0.000E+00	8.310E+04	0.000E+00
SYST_1	5.549E+07	4.075E+07	1.004E+07	1.772E+07
NET	3.071E+07			
AUXILIARY	0.000E+00			
EFFICIENCY	5.534E-01			

SUBSYSTEM: A

CONVERGENCE OF THE INDEPENDENT VARIABLES,
POSSIBLY VERY CLOSE TO THE SOLUTION

OBJECTIVE: 6.16924E+07

VARIABLES

- 1 4.75203E+00 IN_STH.PARM.H
- 2 1.53046E+02 PUMP_SC.EXIT_PRES

CONSTRAINTS

- 1 7.85445E+03 IN_STH.DH=0.0
- 2 4.41247E-10 IN_STH.DP=0.0

**APPENDIX G: LIQUID-METAL
MAGNETOHYDRODYNAMIC POWER PLANT**


```

PROCESS      GP_1:IN
SYSBEG A
PROCESS
  GAS_1-> IN_GAS
  LIQ_1-> IN_LIQ
  GAS_1-> LIQ_1-> MMHD_1 TPNZ_1 SEPR_1 ->GAS_CO ->LIQ_CO
  LIQ_1-> MDIF_1 HT_LIQ MNOZ_1
  GAS_1-> HX_REG:H HT_COOL CP_GAS HX_REG:C
              TPHX_1 <-LIQ_1
              IN_GAS:CYCL
  LIQ_1-> IN_LIQ:CYCL
  NULL-> SYST_1
  VARY MMHD_1.EXIT_PRES = * 8 48 CONS MMHD_1.VOID_FRACTION<0.85
  VARY IN_LIQ.M = * 1.0 450 CONS TPHX_1.VOID_FRACTION>0.55
  VARY TPNZ_1.EXIT_PRES = * 3 47 CONS TPHX_1.PRES_DIFF_IN=0.0
  VARY HT_LIQ.HEAT = * 1E2 5E6 CONS IN_LIQ.DT=0.0
  VARY CP_GAS.EXIT_PRES= * 40 60 CONS IN_GAS.DP=0.0
  VARY HT_COOL.HEAT = * -1E6 0.0 CONS HT_COOL.FLC.TEMP=310.0
  VARY HX_REG.HEAT = * 2E4 4E6
  CONS HX_REG.FLC.TEMP<SEPR_1.FLC1.TEMP-20.0
  MINI -SYST_1.EFFICIENCY
  SWITCH ACC=1E-4 DEL=1E-6 MAXIT=100
SYSEND A
PROCESS
  NULL-> * *:OUT
DATA
  IN_GAS.PARM .ID='JAN-HE'; .T=867; .P=50.0; .V=25.0; .M=1.0;
  IN_LIQ.PARM .ID='LIQ-NA'; .T=867; .P=50.0; .V=25.0; .M=90.0;
  MMHD_1.PARM .EFFICIENCY=0.80; .EXIT_PRES=24.00;
  TPNZ_1.PARM .EFFICIENCY=0.90; .EXIT_PRES=20.0;
  SEPR_1.PARM .VELOCITY_HEAD_RATIO=0.90;
  HX_REG.PARM .HEAT=1E5;
  HT_COOL.PARM .HEAT=-6.5E5;
  HT_LIQ.PARM .HEAT=1.0E6;
  CP_GAS.PARM .EFFICIENCY=0.88; .EXIT_PRES=50.0;
  MDIF_1.PARM .EXIT_VELOCITY=15.0; .EFFICIENCY=0.90;
  MNOZ_1.PARM .EXIT_VELOCITY=25.0; .EFFICIENCY=0.90;
  TPHX_1.PARM .PRES_DROP=0.0;
  SYST_1.PARM .POWER_HEAD_PTR=POWER_HEAD_PTR;
              .FLOW_HEAD_PTR=FLOW_HEAD_PTR;

```

LOOP: A N= 1 NEQ= 4 F= 1.1704E+01
 X= 2.4000E+01 9.0000E+01 2.0000E+01 1.0000E+06 5.0000E+01 -6.5000E+05 1.0000E+05
 C= 6.3200E+00 8.5005E+00 -6.7574E+00 4.0125E+02 -1.8709E-02 2.3848E-01 -2.5742E+02
 LAGRANGE MULTIPLIERS=
 6.2329E+01 -2.0394E-04 7.6832E+01 -2.1107E+01 4.5423E+03 0.0000E+00 4.1462E+00 0.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00
 DISPLACEMENTS=
 1.8296E+00 8.0331E+00 2.5871E-01 3.6624E+05 5.0505E-01 -2.6806E+05 1.8223E+06 -6.2328E-11 1.1102E-15 -7.6833E-11
 2.1259E-11 -4.5423E-09 -4.3201E-12
 HEIGHT = 1.0000E+12
 INITIAL INFEASIBILITY NORM= 2.2742E+05 FINAL INFEASIBILITY NORM= 2.0642E-17

LOOP: A N= 16 NEQ= 4 F= -2.8499E-01
 X= 2.5830E+01 9.8033E+01 2.0259E+01 1.3662E+06 5.0505E+01 -9.1806E+05 1.9223E+06
 C= 7.1630E-01 6.9133E-01 -7.0920E-01 -7.8331E-02 3.3935E-04 1.9890E-01 -7.1593E-01
 L= 4.0652E+02
 LAGRANGE MULTIPLIERS=
 1.8769E+00 -2.4416E-03 2.8131E+00 3.2006E-01 1.4379E+02 0.0000E+00 4.7156E-01 0.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00
 DISPLACEMENTS=
 1.1472E-01 -6.9584E-01 2.4534E-03 -8.8265E+04 -3.2746E-10 -3.7987E+03 -4.5900E+03 -1.8768E-12 2.5258E-15 -2.8131E-12
 -3.2011E-13 -4.7161E-13
 HEIGHT = 1.0000E+12
 INITIAL INFEASIBILITY NORM= 2.0127E+00 FINAL INFEASIBILITY NORM= 1.1761E-23

LOOP: A N= 24 NEQ= 4 F= -2.7898E-01
 X= 2.5944E+01 9.7337E+01 2.0261E+01 1.2780E+06 5.0505E+01 -9.2186E+05 1.9177E+06
 C= 7.4122E-03 -4.5584E-03 3.3785E-05 -4.8639E-04 -4.1536E-06 1.9744E-01 2.5115E-04
 L= 1.3376E-02
 LAGRANGE MULTIPLIERS=
 6.6887E-03 4.8983E-06 2.5450E-02 2.4192E-03 3.5388E-01 0.0000E+00 1.7267E-03 0.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
 0.0000E+00
 DISPLACEMENTS=
 -3.6262E-04 4.5462E-03 9.4921E-05 6.1475E+02 -2.5485E-11 4.9848E+00 5.2545E+00 -6.6890E-15 -1.1276E-17 -2.5449E-14
 -2.4193E-15 -3.5388E-13
 HEIGHT = 1.0000E+12
 INITIAL INFEASIBILITY NORM= 3.2660E-05 FINAL INFEASIBILITY NORM= 1.2593E-25

LOOP: A N= 32 NEQ= 4 F= -2.7900E-01
 X= 2.5944E+01 9.7342E+01 2.0261E+01 1.2786E+06 5.0505E+01 -9.2185E+05 1.9177E+06
 C= -1.2959E-07 -2.0712E-07 1.2836E-09 -2.1673E-08 -9.9880E-11 1.9743E-01 1.0736E-08
 L= 2.2688E-05

IN_GAS

ID=JAN-HE
 TEMP = 8.67000E+02
 PRES = 5.00000E+01
 VEL = 2.50000E+01
 ENTH = 6.05463E+06
 MASS = 1.00000E+00

IN_LIQ

ID=LIQ-NA
 TEMP = 8.67000E+02
 PRES = 5.00000E+01
 VEL = 2.50000E+01
 ENTH = 1.22045E+06
 MASS = 9.73410E+01

MMHD_1

Efficiency = 8.00000E-01
 Power = 1.17465E+06
 VOID FRAC. = 8.50000E-01
 LENGTH = 5.00000E+00
 GRAV ANGLE = 9.00000E+01

TPNZ_1

Efficiency = 9.00000E-01
 VOID FRAC = 8.78581E-01
 SLIP RATIO = 1.00000E+00
 TEMP DIFF = 0.00000E+00
 LENGTH = 5.00000E+00
 GRAV ANGLE = 9.00000E+01

SEPR_1

VEL HEAD RATIO= 9.00000E-01
 PRES DROP(1)= 0.00000E+00
 PRES DROP(2)= 0.00000E+00
 EFFICIENCY= 8.90843E-01
 VAPOR C/O= 0.00000E+00
 HEAT REJECTED= 0.00000E+00
 LIQUID C/O= 0.00000E+00
 GAS C/O= 0.00000E+00

MDIF_1

Efficiency = 9.00000E-01
 LENGTH = 5.00000E+00
 GRAV ANGLE = 9.00000E+01

HT_LIQ

HEAT = 1.27858E+06

HNOZ_1

EFFICIENCY = 9.00000E-01
 EXIT VELOCITY= 2.50000E+01
 LENGTH = 5.00000E+00
 GRAV ANGLE = 9.00000E+01

HX_REG

MODE = DESIGN
 TYPE = COUNTER
 DESIGN MASS FLOW RATES = 1.00 1.00 KG/S
 INLET TEMPERATURES = 856.45 467.40 K
 AVERAGE TEMPERATURES = 671.93 651.93 K
 DESIGN THERMAL RESISTIVITIES = 1.0000E+00 0.0000E+00 0.0000E+00 SQ-M K/W
 OVERALL HEAT TRANSFER COEF = 1.00000E+00 W/SQ-M K
 LOG MEAN TEMP DIFFERENCE = 2.00000E+01 K
 HEAT TRANSFERRED = 1.91774E+06 W
 HEAT TRANSFER SURFACE AREA = 9.58870E+04 SQ-M
 HEAT FLUX = 2.00000E+01 W/SQ-M
 SURFACE TEMPERATURES = 836.45 836.45 K

HT_COOL

HEAT = -9.21854E+05

CP_GAS

MODE = DESIGN
 EXIT PRES = 5.05051E+01
 EFFICIENCY = 8.80000E-01
 MASS FACTOR = 3.94953E-04
 M FACTOR = 1.00000E+00
 PRESSURE RATIO = 2.51787E+00

TPMX_1

void frac. = 7.47428E-01
 SLIP RATIO = 1.00000E+00
 TEMP DIFF = 0.00000E+00
 PRES DROP = 0.00000E+00
 DP_FRAC = 0.00000E+00

OUTPUT BY FLOW

FLOW: GAS_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_GAS	5.000E+01	8.670E+02	2.500E+01	6.055E+06	1.000E+00	3.557E-01	6.055E+06	1.0E+00
MHD_1	2.594E+01	8.595E+02	2.500E+01	6.016E+06	1.000E+00	6.796E-01	6.016E+06	1.0E+00
TPNZ_1	2.026E+01	8.564E+02	9.978E+01	5.999E+06	1.000E+00	8.670E-01	6.004E+06	1.0E+00
SEPR_1	2.026E+01	8.565E+02	9.466E+01	6.000E+06	1.000E+00	8.671E-01	6.004E+06	1.0E+00
HX_REG	2.006E+01	4.874E+02	9.466E+01	4.082E+06	1.000E+00	4.985E-01	4.087E+06	1.0E+00
HT_COOL	2.006E+01	3.100E+02	9.466E+01	3.160E+06	1.000E+00	3.170E-01	3.165E+06	1.0E+00
CP_GAS	5.051E+01	4.674E+02	9.466E+01	3.978E+06	1.000E+00	1.898E-01	3.983E+06	1.0E+00
HX_REG	5.000E+01	8.365E+02	9.466E+01	5.896E+06	1.000E+00	3.432E-01	5.900E+06	1.0E+00
TPHX_1	5.000E+01	8.670E+02	2.500E+01	6.055E+06	1.000E+00	3.557E-01	6.055E+06	1.0E+00

FLOW: LIQ_1

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
IN_LIQ	5.000E+01	8.670E+02	2.500E+01	1.220E+06	9.734E+01	1.235E-03	1.188E+08	0.0E+00
MHD_1	2.594E+01	8.595E+02	2.500E+01	1.209E+06	9.734E+01	1.232E-03	1.177E+08	0.0E+00
TPNZ_1	2.026E+01	8.564E+02	9.978E+01	1.204E+06	9.734E+01	1.231E-03	1.177E+08	0.0E+00
SEPR_1	2.026E+01	8.567E+02	9.466E+01	1.205E+06	9.734E+01	1.231E-03	1.177E+08	0.0E+00
HDF_1	5.178E+01	8.578E+02	1.500E+01	1.209E+06	9.734E+01	1.231E-03	1.177E+08	0.0E+00
HT_LIQ	5.178E+01	8.683E+02	1.500E+01	1.222E+06	9.734E+01	1.235E-03	1.190E+08	0.0E+00
MHDZ_1	5.000E+01	8.683E+02	2.500E+01	1.222E+06	9.734E+01	1.235E-03	1.190E+08	0.0E+00
TPHX_1	5.000E+01	8.670E+02	2.500E+01	1.220E+06	9.734E+01	1.235E-03	1.188E+08	0.0E+00

FLOW: GAS_CO

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
SEPR_1	2.026E+01	8.565E+02	9.466E+01	6.000E+06	0.000E+00	8.671E-01	0.000E+00	1.0E+00

FLOW: LIQ_CO

MODEL	PRES. (ATM)	TEMP. (K)	VELOCITY (M/S)	ENTH. (J/KG)	MASS (KG/S)	SPEC VOL (M**3/KG)	ENERGY (W)	QUALITY
SEPR_1	2.026E+01	8.567E+02	9.466E+01	1.205E+06	0.000E+00	1.231E-03	0.000E+00	0.0E+00

POWER SUMMARY

MODEL	INPUT (W)	PRODUCED (W)	CONSUMED (W)	LOSS (W)
IN_GAS	-5.898E+02	0.000E+00	0.000E+00	0.000E+00
IN_LIQ	2.535E+02	0.000E+00	0.000E+00	0.000E+00
MHD_1	0.000E+00	1.175E+06	0.000E+00	0.000E+00
SEPR_1	0.000E+00	0.000E+00	0.000E+00	0.000E+00
HT_LIQ	1.279E+06	0.000E+00	0.000E+00	0.000E+00
HT_COOL	0.000E+00	0.000E+00	0.000E+00	9.219E+05
CP_GAS	0.000E+00	0.000E+00	8.179E+05	0.000E+00
SYST_1	1.279E+06	1.175E+06	8.179E+05	9.219E+05
NET	3.567E+05			
AUXILIARY	0.000E+00			
EFFICIENCY	2.790E-01			

SUBSYSTEM: A

NORMAL TERMINATION

OBJECTIVE: -2.79003E-01

VARIABLES

- 1 2.59439E+01 MHD_1.EXIT_PRES
- 2 9.73418E+01 IN_LIQ.M
- 3 2.02613E+01 TPNZ_1.EXIT_PRES
- 4 1.27053E+06 HT_LIQ.HEAT
- 5 5.05051E+01 CP_GAS.EXIT_PRES
- 6 -9.21054E+05 HT_COOL.HEAT
- 7 1.91774E+06 HX_REG.HEAT

CONSTRAINTS

- 1 -9.93804E-11 MHD_1.VOID_FRACTION<0.85
- 2 1.97428E-01 TPHX_1.VOID_FRACTION>0.55
- 1 -1.29536E-07 TPHX_1.PRES_DIFF_IN=0.0
- 2 -2.07116E-07 IN_LIQ.DT=0.0
- 3 1.28365E-09 IN_GAS.DP=0.0
- 4 -2.16734E-08 HT_COOL.FLC.TEMP=310.0
- 3 1.07357E-08 HX_REG.FLC.TEMP<SEPR_1.FLC1.TEMP-20.0

Distribution for ANL/FE-85-3

Internal:

J.G. Asbury	S.J. Grammel	C.V. Pearson
F.C. Bennett	W. Harrison	M. Petrick
M.J. Bernard	J.E. Helt	G.N. Reddy
G.K. Berry (33)	D.R. Henley	J.J. Roberts
S.K. Bhattacharyya	H.S. Huang	N.F. Sather
D.J. Bingham	J.F. Koenig	W.W. Schertz
L.W. Carlson	M. Krumpelt	R. Sekar
K.C. Chang	K.D. Kuczen	Y.W. Shin
S.U. Choi	J. Lazar	A.J. Sistino
L.S. Chow	C. Lee	T.G. Surles
J.M. Cook	G.P. Lewis	C.E. Swietlik
E.J. Croke	R.A. Lewis	A. Thomas
E.J. Daniels	C.D. Livengood	S.P. Vanka
E.M. Dean	R.W. Lyczkowski	C.S. Wang
C.B. Dennis	K.S. Macal	A.M. Wolsky
D.R. Diercks	V. Minkov	ANL Contract Copy
J.J. Dzingel	K.M. Myles	ANL Libraries (2)
H.K. Geyer (10)	O.O. Ohlsson	ANL Patent Department
R.F. Giese	C.B. Panchal	TIS Files (6)

External:

U.S. Department of Energy Technical Information Center, for distribution per UC-32 and UC-90 (242)

Manager, U.S. Department of Energy Chicago Operations Office (DOE-CH)

Energy and Environmental Systems Division Review Committee:

R.S. Berry, The University of Chicago

G.E. Dials, Dials and Assoc., Santa Fe, N.M.

B.A. Egan, Environmental Research and Technology, Inc., Concord, Mass.

W.H. Esselman, Electric Power Research Institute, Palo Alto, Calif.

M.H. Kohler, Bechtel National, Inc., San Francisco

J.W. McKie, University of Texas, Austin

N.C. Mullins, Virginia Polytechnic Institute and State University, Blacksburg

J.J. Stukel, University of Illinois, Urbana

J.J. Wortman, North Carolina State University, Raleigh

R.D. Andrews, Rocky Mountain Energy, Broomfield, Colo.

R. Bajura, Morgantown Energy Technology Center, U.S. Department of Energy, Morgantown, W.Va.

J.M. Begovich, Oak Ridge National Laboratory, Oak Ridge, Tenn.

S.K. Beer, Morgantown Energy Technology Center, U.S. Department of Energy, Morgantown, W.Va.

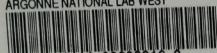
S. Biondo, Office of Fossil Energy, U.S. Department of Energy, Washington, D.C.

H.H. Blecker, ICARUS Corp., Rockville, Md.
 D.P. Bloomfield, PSI/Systems, Andover, Mass.
 A. Boni, Physical Sciences, Inc., Andover, Mass.
 H. Branover, Ben-Gurion University, Tel Aviv, Israel
 D. L. Breton, Dow Chemical USA, Plaquemine, La.
 R. Carabatta, Pittsburgh Energy Technology Center, U.S. Department of Energy,
 Pittsburgh
 P. Chung, University of Illinois, Chicago
 J.G. Cleland, Research Triangle Institute, Research Triangle Park, N.C.
 K. Craig, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 J. Cutting, Gilbert Associates, Reading, Penn.
 K.J. Daniel, General Electric Corporate R&D, Schenectady, N.Y.
 S. Divakaruni, Electric Power Research Institute, Palo Alto, Calif.
 J.S. Dweck, J.S. Dweck, Inc., Denver, Colo.
 A. Dyson, Tennessee Valley Authority, Chattanooga, Tenn.
 J. Elliott, Massachusetts Institute of Technology, Cambridge, Mass.
 G. Enyedy, PDQ\$, Inc., Gates Mills, Ohio
 M. Faist, Radian Corp., Austin, Texas
 L.T. Fan, Kansas State University
 J. Fillo, Environmental Research and Technology, Inc., Pittsburgh
 J. Fisher, Stone and Webster, Boston
 H.J. Gadiyar, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 P.W. Gallier, ASPEN Technology, Inc., Cambridge, Mass.
 G. Garrison, University of Tennessee Space Institute, Tullahoma, Tenn.
 E.W. Geller, Flow Industries, Inc., Kent, Wash.
 J.H. Gibbons, Office of Technology Assessment, U.S. Congress
 F.D. Gmeindl, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 L.E. Graham, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown W.Va.
 H. Hagler, Hagler, Bailly, and Co., Washington, D.C.
 K. Haynes, Foster Wheeler Synfuels Corp., Livingston, N.J.
 J. Henry, University of Tennessee at Chattanooga
 S.C. Hill, Los Alamos National Laboratory, Los Alamos, N.M.
 R. Holmann, Westinghouse Electric Corp., Pittsburgh
 F. Honea, Grand Forks Project Office, Grand Forks, N.D.
 D.A. Horazak, Westinghouse Electric Corp., Concordville, Penn.
 W. Jackson, HNJ Corp., Washington, D.C.
 B. Joseph, Washington University, St. Louis, Mo.
 D.E. Kash, University of Oklahoma
 A.A. Khan, Union Carbide Corp., Oak Ridge, Tenn.
 S. Knoke, Flow Industries, Inc., Kent, Wash.
 D. Krastman, Pittsburgh Energy Technology Center, U.S. Department of Energy,
 Pittsburgh
 H. Link, Solar Energy Research Institute, Golden, Colo.
 T. Littert, Westinghouse R&D, Pittsburgh

P.S. Lowell, P.S. Lowell Co., Inc., Austin, Texas
 C. Mah, Aerojet Energy Conversion Co., Sacramento, Calif.
 T. McCloskey, Notre Dame College, South Euclid, Ohio
 W.J. McMichael, Research Triangle Institute, Research Triangle Park, N.C.
 M.C. Millman, Halcon Computer Technology, New York
 L. Miller, U.S. Department of Energy, Germantown, Md.
 L. Mims, Chicago
 L.M. Naphtali, U.S. Department of Energy, Washington, D.C.
 S.A. Newman, Foster Wheeler Energy Corp., Livingston, N.J.
 J. Notestein, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 T. O'Brien, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 A. Pappano, Pasadena, Calif.
 M. Paskin, Allison Gas Turbine, Indianapolis, Ind.
 J. Patten, Gilbert Associates, Reading, Penn.
 L. Perini, Applied Physics Lab, Johns Hopkins Laboratory, Laurel, Md.
 M. Perlmutter, U.S. Department of Energy, Pittsburgh
 T.T. Philips, Los Alamos National Laboratory, Los Alamos, N.M.
 R. Piccirelli, Wayne State University, Detroit, Mich.
 E. Pierson, Purdue University-Calumet, Hammond, Ind.
 A.A. Pitrolo, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 P. Probert, Babcock and Wilcox Co., Barberton, Ohio
 G.H. Quentin, Electric Power Research Institute, Palo Alto, Calif.
 R. Raghavan, Foster Wheeler Energy Corp., Livingston, N.J.
 M.W. Reed, Tennessee Valley Authority, Chattanooga, Tenn.
 I.H. Rinard, Halcon SD Group, New York
 L. Saroff, Dravo Engineers, Inc., Pittsburgh
 R. Shinnar, City College of New York
 C.H. Sink, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 D.P. Smith, General Electric Corporate R&D, Schenectady, N.Y.
 I. Smith, The City University, London, U.K.
 G. Steinfeld, Science Applications, Inc., Morgantown, W.Va.
 S.S. Stern, Halcon SD Group, New York
 K. Stone, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 B. Svrcak, University of Calgary, Alberta
 D. Swink, Office of Fossil Energy, U.S. Department of Energy, Washington, D.C.
 J. Templemeyer, Southern Illinois University, Carbondale, Ill.
 W.C. Thomas, Radian Corp., Austin, Texas
 W. Trzaskoma, Gilbert Assoc., Inc., Reading, Penn.
 V.S. Underkoffler, Gilbert Assoc., Inc., Reading, Penn.
 S.R. Vatcha, Ashland Oil, Inc., Ashland, Ky.
 K. Vyas, Morgantown Energy Technology Center, U.S. Department of Energy,
 Morgantown, W.Va.
 R.E. Weinstein, Gilbert/Commonwealth, Reading, Penn.

J. Weisman, University of Cincinnati, Ohio
W. Wells, Center for Research on Sulfur in Coal, Champaign, Ill.
G. Wheeler, U.S. Department of Energy, Germantown, Md.
F. Wong, Electric Power Research Institute, Palo Alto, Calif.
S. Wu, University of Tennessee Space Institute, Tullahoma, Tenn.
R.K. Young, Stearns-Catalytic, Inc., Homer City, Penn.
J. Zaranek, U.S. Steel Corp., Monroeville, Penn.

ARGONNE NATIONAL LAB WEST



3 4444 00008916 9

X

